Universidad Carlos III de Madrid

www.uc3m.es

LOCALIZACIÓN DE ROBOTS MÓVILES EN ENTORNOS TRIDIMENSIONALES MEDIANTE VISIÓN ESTÉREO Y CUDA

Proyecto Final de carrera

Departamento de Ingeniería de Sistemas y Automática

13 DE OCTUBRE DE 2014

Autor: Miguel Díez-Ochoa Díez

Tutor: Juan Carlos González Víctores

Titulación: Ingeniería Industrial

Lugar: Escuela Politécnica Superior Carlos III, Madrid

Título: Localización de robots móviles en entornos tridimensionales mediante visión estéreo y CUDA
Autor: Miguel Díez-Ochoa Díez
Tutor: Juan Carlos González Víctores

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

A mi familia, que tanto apoyo y ayuda me ha ofrecido.
A los compañeros, que tantos momentos hemos compartido durante las largas temporadas de biblioteca.
A los profesores que no finalizan su labor al terminar la clase.
Y a la persona que me ha enseñado que cualquier problema se puede resolver tirando de un simple lápiz.
Gracias.

ÍNDICE

ÍNDICE DE FIGURAS

RESUMEN

En este documento se ha desarrollado un método de localización para robots móviles mediante visión por computación. Este método utiliza la información proporcionada por dos cámaras, y un sistema odométrico compuesto por un acelerómetro y un giróscopo (y/o brújula de tres ejes).

El método se desarrolla en dos pasos. El primer paso consiste en la obtención de un modelo del entorno, creando un mapa tridimensional. El algoritmo empleado para la obtención de este modelo, Iterative Closest Points (ICP), es usado para la reconstrucción de superficies tridimensionales. La reconstrucción puede considerarse un proceso que no impone requisitos temporales. Este mapa sólo necesita generarse una vez, previa a la realización de tareas por parte del robot. El modelo obtenido se almacena, y es el que se emplea para el segundo paso.

El segundo paso está orientado a realizarse durante el funcionamiento normal del robot: se trata de la localización del robot dentro del modelo obtenido previamente. Para ello, el método compara las porciones del entorno que las cámaras son capaces de visualizar con el modelo. Se obtienen valores de correspondencia para las posibles localizaciones.

Debido a que el tratamiento de información tridimensional requiere procesar un alto volumen de información, se ha optado por hacer uso de la tarjeta gráfica para optimizar el tiempo de cómputo. Específicamente, se ha optado por la utilización de una tarjeta gráfica de NVIDIA, para hacer uso de su lenguaje de programación CUDA. Este lenguaje ha demostrado obtener mejores resultados que otros lenguajes de programación que sirven para programar mayor variedad de tarjetas gráficas.

ABSTRACT

In this document, a localization method for mobile robots based on computer vision has been developed. This method uses the information provided by two cameras and an odometric system formed by an accelerometer and a gyroscope (a three axis compass can also be used).

This method is built in two steps. The first one consists on obtaining a model of the environment, creating a three dimensional map. The algorithm used to obtain the model is named Iterative Closest Points (ICP). It is used to match different surfaces to reconstruct the environment. The reconstruction of the environment is a process that only needs to be executed once, previous to the release of task by the robot. The model is saved and it is used by the second step to achieve its goal.

The second step is planned to be performed during the normal operation of the robot. The goal is the obtaining of the localization of the robot inside the model that has been obtained in the first step. For this, the method compares the portion of the environment that is in the field vision of the robot with the model of the map. For each comparison the algorithm provides the different possible localizations.

Because of the treatment of the high volume of data to process of the three dimensional information, the choice of using an Nvidia GPU and using their programming language, CUDA, has been taken. This programming language has proved having better results than other multiplatform programming languages.

1. INTRODUCCIÓN

El campo de la localización en robótica y vehículos autónomos está experimentando un elevado crecimiento debido a las nuevas necesidades de la sociedad y de las industrias. Por ello, este campo está recibiendo grandes inversiones para poder satisfacer dicha demanda. El caso más común puede ser el de la navegación por satélite GNSS [1], que actualmente está siendo explotado por los servicios NAVSTAR-GPS de los Estados Unidos y GLONASS de Rusia[1], además de otros que están actualmente en desarrollo, como son el sistema Galileo, proyecto de la Unión Europea, y el Beidou, perteneciente a la República Popular de China.

1.1. Motivación

Los sistemas basados en las constelaciones de satélites resultan realmente eficaces para localizar dispositivos en zonas con una buena calidad de señal, como exteriores, tanto tierra, mar o aire. Son también útiles en entornos urbanos, aunque pierden precisión debido a los obstáculos, como es el caso de los edificios. Carecen de utilidad en aquellos lugares con una pobre cobertura como pueden ser el interior de instalaciones o viviendas, entornos subterráneos, subacuáticos y espaciales (Figura 1).

A raíz de esas limitaciones surgen las necesidades de encontrar métodos alternativos que puedan proporcionar un servicio semejante. Actualmente hay desarrollados diversos métodos que satisfacen estas necesidades. Entre estos métodos se pueden encontrar aquellos que hacen uso de marcadores codificados, señales electromagnéticas, o información del entorno. Algunos de los métodos anteriormente citados tienen el inconveniente de necesitar una infraestructura en la que apoyarse, resultando por ello menos flexibles ante cambios en la disposición del entorno. En los últimos años una solución a dicho problema que ha logrado posicionarse como una opción ventajosa frente al resto ha sido la localización por Wi-Fi. Sin embargo, este método requiere la existencia de una infraestructura Wi-Fi. Debido a las limitaciones citadas, se ha querido desarrollar un método de localización que sea capaz de superar dichas limitaciones.

Figura 1: Recreación artística del rover Curiosity proporcionada por la NASA

[1] Desarrollado por la Unión Soviética y actualmente gestionado por Rusia

1.2. Objetivos

Debido a los problemas que se han expuesto anteriormente, es preciso desarrollar un método capaz de proporcionar el servicio con una calidad acorde a nuestras necesidades. Especialmente en el campo de la robótica, que tanto auge está experimentando en los últimos años, se harán necesarias mayores precisiones en la localización debido a las exigencias requeridas de la tarea a realizar y a la peligrosidad que pueda existir.

Por ello, en este documento se ha diseñado un método que responda ante las limitaciones anteriormente descritas, eliminando la necesidad de tener una infraestructura de soporte, que pueda adaptarse a distintos entornos con un menor coste y que disponga de la precisión necesaria para el desarrollo de la tarea.

Como se ha explicado anteriormente en el resumen del documento, se va a hacer uso de dos de cámaras para obtener el mapa de disparidad de una porción del entorno. A partir de ese mapa de disparidad, podremos obtener el mapa de profundidad. Estos pasos serán explicados más adelante. Este mapa de profundidad representa la distancia desde el par de cámaras hasta los elementos que conforman el entorno. Este proceso es análogo al funcionamiento de la visión humana, que mediante las diferencias entre las imágenes obtenidas por cada ojo estimamos de forma inconsciente las distancias a los objetos.

Para crear un modelo tridimensional del entorno, deberemos ser capaces de superponer varios mapas de profundidad entre sí, enlazando los puntos comunes que existan entre los distintos mapas y añadiendo al modelo del entorno los puntos nuevos. Esto es realizado durante el primer paso del método propuesto.

Una vez finalizada la recreación del modelo del entorno, para localizar el robot, se realiza una comparación de los mapas de profundidad realizados por él con el modelo tridimensional realizado en el paso previo. Para facilitar el seguimiento, se podrá hacer uso de técnicas de odometría y restar carga computacional al sistema. Podrán ser tanto técnicas de odometría visual, inercial o por estimación odométrica.

1.3. Estructura del documento

A lo largo de este documento se tratan los siguientes apartados. Primeramente, en el capítulo 2, se introducen los aspectos más relevantes del método desarrollado, realizando una introducción a los sistemas de visión y de localización. Con la base puesta en estos 2 sistemas se procede a introducir un concepto que unifica ambas técnicas, la de localización y la de visión, como es el de los sistemas de modelado de entornos basados en visión. Y con relación a los sistemas de modelado de entornos, tenemos los métodos de localización. Un último apartado del capítulo 2 lo dedicamos para introducir el lenguaje de programación CUDA.

El capítulo 3 de este documento es el perteneciente a la arquitectura global del sistema. En él se explica la organización del programa desarrollado desde su nivel más conceptual a uno más detallado, detallando el funcionamiento de los procesos y funciones involucradas en la ejecución de dicho método.

En el capítulo 4 se proporciona el conocimiento técnico que rige los procesos que se ven involucrados en el documento, en la implementación y diseño de éste método. Éstas son las ecuaciones de la óptica de las cámaras, el algoritmo ICP, estimación odométrica…

En el capítulo 5, se exponen los resultados obtenidos durante la fase de pruebas del método y se comentan dichos resultados. Y por último, en el capítulo 6, se recogen las conclusiones obtenidas de los resultados mostrados en el capítulo anterior y posibles líneas para futuras investigaciones que puedan seguirse para conseguir una mejora de los sistemas de localización de robots.

2. ESTADO DEL ARTE

En este capítulo se realiza un resumen sobre la actual situación de los sistemas de localización y de reconstrucción de entornos. Para ello, se comenzará desde aquellos medios que permiten lograr dichos objetivos, como son la visión por computación y los sistemas odométricos, ya que son las bases en las que está cimentado este método.

2.1. Visión artificial, tratamiento de imágenes

Se define visión artificial como el análisis de las imágenes por medio de computadores para extraer la información requerida de ellas y obtener conocimiento sobre los objetos físicos, como pueden ser medidas, texturas, formas... Este campo de la ciencia surge de la necesidad de dotar a ciertos sistemas de las capacidades propias de un humano, como es la visión, ya que resultan imprescindibles para le realización de múltiples tareas [2], pero eliminando los inconvenientes que conlleva que sean realizadas por un ser humano. Los pasos a tener en cuenta son:

2.1.1. Adquisición y digitalización de la imagen

Conversión de ondas electromagnéticas incidentes sobre el sensor en señales eléctricas procesables por un ordenador. Para ello es necesario disponer de un sensor que capte la información, una óptica que converja los haces luminosos en el sensor, un sistema de iluminación adecuado que permita simplificar el problema a tratar y en algunos casos una tarjeta de adquisición de imágenes.

2.1.2. Sistema de procesamiento de imágenes

En la visión humana, se estima que el 75% de la información que procesa el cerebro proviene del sentido de la vista, por lo que en la visión artificial la capacidad de cómputo ha venido siendo un limitante importante para el desarrollo de este campo. Se hace uso de procesadores cada vez con mayor potencia de cálculo para poder extraer la información requerida de las imágenes en menor tiempo, como es el caso del uso de tarjetas gráficas en el procesado de imágenes.

2.1.3. Medio de salida

La información obtenida por medio del procesamiento de las imágenes tiene que proporcionar alguna utilidad, por lo que se necesita que algún dispositivo muestre, almacene o procese dicha información. Estas salidas pueden ser monitores y que un operario visualice los resultados, o también pueden ser autómatas que tengan que modificar su comportamiento para recoger un objeto a través de las coordenadas que el procesamiento de imágenes le ha proporcionado.

2.1.4. Aplicaciones

Las aplicaciones de la visión por computación son múltiples, aunque quizás la más extendida y la que mayores esfuerzos han dedicado en su implantación es la fabricación y en el control de calidad. Con la instalación de un sistema de visión por computación en una industria se pretendía conseguir una mejora en los sistemas de inspección de los productos, cambiando de unos métodos estadísticos en base a los análisis de una muestra y extrapolándolos a una población mayor a unos controles en los que se analiza toda la producción de forma automatizada. En la industria militar ha cobrado gran importancia debido a la implantación de sistemas de visión a determinado tipo de armamento o a los sistemas de inteligencia por análisis de imágenes. Otros tipos de aplicaciones de los sistemas de visión por computación son:

a. Ocio
b. Medicina
c. Vigilancia y seguridad
d. Reconocimiento de formas
e. Reconocimiento gestos

2.2. Sistemas visuales de medición de distancias

Los sistemas visuales de medición de distancias son aquellos que hacen uso de las propiedades de las ondas electromagnéticas para realizar los cálculos oportunos que proporcionen una magnitud correspondiente a la distancia de un objeto al sensor.

2.2.1. Visión estéreo

Mediante este sistema se es capaz de combinar la información que proporcionan las imágenes que provienen de diferentes perspectivas para calcular la distancia a un objeto. Para ello se necesitan conocer los parámetros constructivos del par estéreo y los parámetros intrínsecos de cada una de las cámaras. El fundamento teórico de este sistema se basa en que la proyección de la imagen de un objeto no incide en la misma posición en ambos sensores, pudiendo entonces calcular la distancia por medio de dichas diferencias en las proyecciones.

2.2.2. Láser

El principio de funcionamiento de estos dispositivos se basa en la diferencia entre las fases de 2 ondas. La primera, y la que se toma como referencia, es la onda emitida, que una vez que la onda incide sobre un cuerpo es reflejada y recibida por un sensor. Debido al tiempo que transcurre entre que la primera onda es emitida y que la reflexión de ella es recibida, las fases de ambas ondas son diferentes. Conocido el desfase entre las ondas y la velocidad de propagación de la onda en el medio se conoce la distancia.

2.2.3. Cámaras Time-of-flight

Las cámaras time-of-flight o tiempo de vuelo utilizan el mismo principio que las del método anterior. Hacen uso del tiempo que tarda una onda en viajar desde el emisor hasta el receptor. La diferencia entre el láser y las time-of-flight radica en que las últimas miden directamente el tiempo transcurrido en vez del desfase producido por realizar ese trayecto. Para ello, se hacen uso de cámaras precisión de 1/10^10 segundos (este caso es particular para la Kinect v2).

2.2.4. Luz estructurada

Se hace uso de un patrón de luz conocido para calcular distancias. Esta operación se realiza en función de la posición donde se proyecta el patrón sobre dicho sensor. La magnitud medida es el ángulo con el que incide el patrón sobre el sensor. Para ello se debe conocer el ángulo con el que se emite cada patrón, y la distancia entre el emisor del patrón y el receptor.

2.3. Odometría

La palabra odometría proviene del griego, de la unión de las palabras "*hodos*" y *"metro"*, que significan camino y medir respectivamente. Actualmente se conoce como "odometría" a las técnicas de localización basadas en la información proporcionada por sensores exteroceptivos[2] como sensores de visión y sensores inerciales, y propiocetivos[3], como los encoders. Éstos permiten obtener una estimación de la posición real en la que se encuentra un sistema, en un

[2] Aquellos que adquieren datos del exterior sistema
[3] Aquellos que adquieren datos del propio sistema

determinado instante de tiempo, respecto a un sistema de referencia inicial. Según la técnica utilizada pueden clasificarse en los siguientes grupos:

2.3.1. Estimación odométrica

Esta técnica se basa en el cálculo de la posición y de la trayectoria de un objeto mediante el conocimiento de su velocidad y dirección. Los orígenes de esta técnica se remontan a la navegación por estima[4]. Actualmente, las configuraciones de tracción diferencial, permiten calcular la posición del robot a partir de las ecuaciones geométricas, que surgen de la relación entre los componentes del sistema de propulsión y de los encoders [3].

El encoder es un dispositivo electromecánico que se compone de un disco perforado solidario a un eje de rotación, un foto emisor y un foto receptor que se activa cuando la luz atraviesa alguna de las ranuras del disco. Existen 2 modelos de encoder. El absoluto, que tiene el disco codificado con '*n*' pistas, dividiendo la circunferencia en 2^n posiciones (Figura 2). Su uso general es la medida de magnitudes angulares. Y el encoder incremental (Figura 3), más extendido para medir desplazamientos lineales que el caso anterior, dispone de dos señales en cuadratura para permitir determinar el sentido de giro.

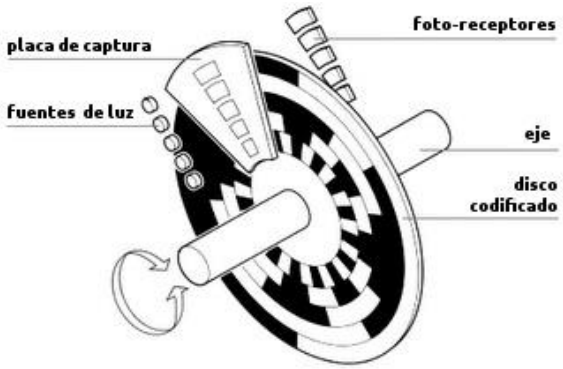

Figura 2: Encoder absoluto

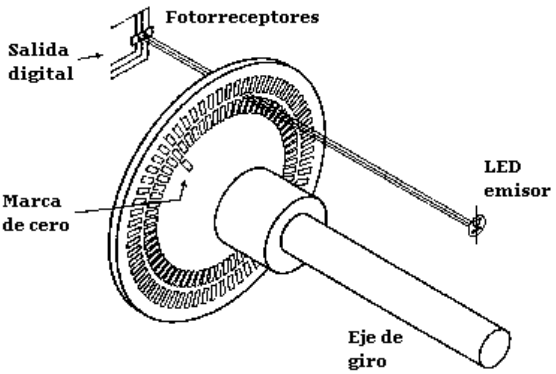

Figura 3: Encoder relativo

[4] La navegación a estima es una técnica mediante la cual por medios analíticos y a partir de la ubicación inicial, velocidad y dirección se estima la trayectoria y la ubicación.

Los foto-receptores pueden ser sustituidos por sensores Hall y los foto emisores por imanes. Si acoplamos los imanes al disco giratorio y los sensores Hall fijos al estator, se denomina configuración de imán móvil, si por el contrario tanto el imán como el sensor están fijos en el estator y es el disco perforado el que crea las perturbaciones en el campo magnético obtenemos una configuración de imán fijo. Hay otros métodos de calcular posiciones como pueden ser métodos resistivos, inductivos o capacitivos, quedando éstos más relegados a movimientos de menor rango, al igual que el encoder absoluto. Las aplicaciones de esta técnica odométrica han sido tanto militares como civiles.

a. Medición de terrenos
b. Control de robots móviles
c. Contador kilómetros de coches
d. Navegación de naves.

2.3.2. Navegación inercial

Los orígenes de las técnicas de navegación inercial son relativamente modernas, no nacen hasta el siglo XX. Tienen como antecedentes la navegación por estima y pudieron desarrollarse gracias al giróscopo, inventado por Leon Focault en el siglo anterior. Un Sistema de Navegación Inercial (INS) consiste en la determinación de las componentes de la posición, dirección y velocidad de un móvil mediante el tratamiento de las fuerzas inerciales que actúan sobre él.

Se pueden subdividir en dos tipos los INS

- El sistema gimbaled aísla la plataforma con los sensores inerciales de los movimientos de rotación externos. Estas plataformas están sujetas a un marco que rota de tal manera que aísla el interior de las rotaciones externas.
- El sistema strap-down tiene alineados los ejes de los sensores con los ejes del móvil en todo momento. Este método es patentado en 1956.

Durante la II Guerra Mundial fueron ampliamente utilizados estos sistemas para el guiado de misiles. Siendo desarrollado este sistema de guiado por Robert Goddard. Los sistemas de navegación inercial se han utilizado tanto en vehículos terrestres, aéreos, marinos y submarinos. Gracias a estos sistemas, el submarino USS Nautilus cruzo el polo Norte bajo el hielo en 1958. En aviónica está permite el uso de los sistemas inerciales como Sistema de Navegación como Medio[5] Único (Sole-Means Navigation System) a diferencia de los sistemas GPS, debido a que no proporcionan la cobertura, disponibilidad e integridad que se requeridas para su utilización como sistema de navegación como Medio Único.

Para describir la trayectoria de un móvil con precisión son necesarias como mínimo unas medidas de movimiento lineal y otras de movimiento angular. Para aquellas medidas de movimiento lineal, se hace uso de los acelerómetros, dispositivos que miden las aceleraciones sufridas, y para aquellos movimientos angulares se disponen de brújulas o de giróscopos, proporcionando una magnitud en función de la dirección y de la velocidad angular respectivamente. Se pueden clasificar estos sensores por los principios físicos que influyen en la medición. Para los acelerómetros tenemos la siguiente clasificación:

[5] Un sistema de navegación aprobado como medio único para determinada operación o fase del vuelo debe posibilitar a la aeronave satisfacer, en dicha operación o fase del vuelo, los cuatro requisitos de actuación del sistema de navegación: precisión, integridad, disponibilidad y continuidad del servicio.

1. Mecánicos: Son dispositivos inerciales, por medio de la segunda ley de Newton, son capaces de calcular la aceleración que sufre el móvil midiendo la fuerza de inercia de una masa interna. Las masas deforman las galgas extensométricas lo que proporciona la magnitud de la aceleración.
2. Piezoeléctricos: las aceleraciones del móvil provocan una deformación física de un material que genera una variación en la polarización eléctrica. Apareciendo una diferencia de potencial y cargas eléctricas en la superficie del mismo.
3. Piezoresistivos: El funcionamiento es similar al del piezoeléctrico, difiriendo en que el piezoresistivo modifica su conductividad al ser sometido a esfuerzos en vez de crear diferencias potenciales.
4. Capacitivos: Las aceleraciones producen un desplazamiento relativo de una de las placas del condensador modificando su capacidad.
5. Térmicos: Una resistencia con dos termopares en los extremos es introducida en el interior de una placa de silicio que actúa como calefactor. De esta forma se consigue formar una cavidad de aire caliente, llamada burbuja, sobre los termopares. Las aceleraciones que sufre la burbuja y su gradiente de temperatura provocan que la región caliente se desplace en el sentido de la aceleración, modificando los parámetros de los termopares.

Clasificación giróscopos:
1. Giróscopo mecánico: Según la ley de conservación del momento angular, una masa que gira sobre un eje experimenta una momento opuesto a aquellos que intentan causar una modificación de su momento angular, en este caso, el momento experimentado se opone al cambio del eje de rotación de la masa (Figura 4).
2. Giróscopo de estructura vibrante: estos dispositivos se caracterizan por disponer de un elemento vibrante en su interior que al ser rotado, es afectado por la fuerza de Coriolis y genera una segunda vibración perpendicular a la original [4]. Las diferentes arquitecturas giróscopos que podemos encontrar son:

- Piezoeléctrico
- Copa de vino resonador
- Vibrador cilíndrico
- Tenedor
- De rueda

3. Giróscopo óptico: están basados en un principio llamado efecto Sagnac[6]. El principio de este efecto consiste en que si dos haces de luz se mueven en direcciones opuestas y el sistema sufre un movimiento de rotación, el haz que se mueve en sentido opuesto a dicha rotación recorrerá el circuito en menor tiempo [5].

- Laser: por medio de espejos se forma un circuito cerrado.
- De fibra óptica: se dispone de un circuito de fibra óptica por la cual se mueven los haces.

[6] El efecto Sagnac se consiste en que *"una onda electromagnética que se mueve en un camino cerrado, que está rodeada por un área finita, es influenciada por la velocidad angular en la cual está incluida esa área"*

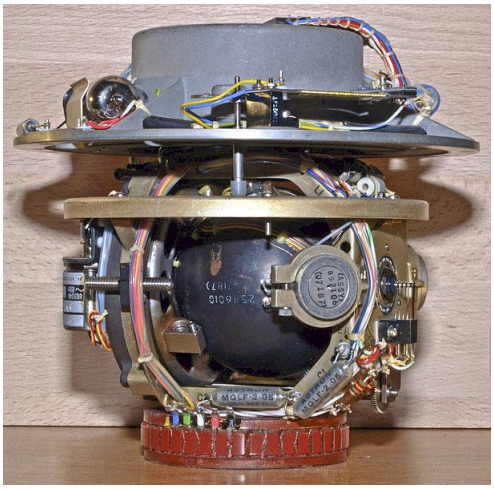

Figura 4: Giróscopo direccional, Sperry Phoenix Co.

A continuación se muestran algunas aplicaciones de los sensores inerciales:

a. Navegación
b. Ayudas para mejorar localización GNSS, Wi-Fi…
c. Seguridad de automóviles: control de estabilidad, airbag
d. Entretenimiento: consolas, móviles, tablets…
e. Fotografía: estabilizador óptico de imagen
f. Robótica: mantener equilibrio, estimar recorrido, segway…
g. Geología: sismología

2.3.3. Odometría visual

Se puede definir la odometría visual como el proceso que permite obtener la ubicación de una cámara o de un sistema de cámaras mediante el análisis de imágenes, sin ser necesario ningún conocimiento del entorno. Recientemente, los sistemas basados en visión artificial se han convertido en un elemento muy importante dentro del desarrollo de sistemas inteligentes. Los principales motivos son un menor coste, mayor rapidez y menor consumo energético que otros sistemas que realizan tareas similares como pueden ser los sistemas LASER.

En el campo de la navegación se han desarrollado algoritmos que pueden estimar y detectar con precisión las trayectorias seguidas por un móvil [6]. Estos móviles pueden ser desde robots de pequeñas dimensiones a vehículos totalmente independientes. Para poder realizar una estimación de los parámetros del movimiento, se deben obtener en un primer paso unos puntos de referencia, durante un proceso posterior, se eliminan aquellos puntos que no sean estacionarios, tomando como muestra solo aquellos puntos de referencia de las imágenes que sean estacionarios. Esto se debe al hecho de no poder estimar con precisión la posición de nuestro sistema si la estamos referenciando respecto a otro sistema móvil cuya cinemática es desconocida. Para ello, se han desarrollado algoritmos que sean capaces de hacer dicha distinción como puede ser el algoritmo RANSAC u otros modelos probabilísticos.

En el campo de la navegación el sistema debe ser capaz de tratar toda la información en tiempo real, por lo que es realmente importante tener unos algoritmos bien implementados así como un hardware que pueda soportarlo.

2.3.3.1. Vehículos inteligentes

Los vehículos inteligentes son aquellos que incorporan los sistemas de percepción, con los cuales son capaces de obtener información tanto del entorno como de aspectos internos del vehículo, y los sistemas actuadores, que permitan modificar algunos aspectos del comportamiento del vehículo adaptándolo a las circunstancias requeridas. La tendencia de estos vehículos es la navegación autónoma en los entornos habituales, para así poder aumentar la seguridad vial disminuyendo la relevancia de aquellos elementos que puedan suponer un factor de peligrosidad, como es la fatiga o la somnolencia del conductor. Otro elemento a tener en cuenta es del ahorro de combustible, haciendo que dichos vehículos tengan un modelo de conducción más eficaz, eliminando las formas de conducciones que provocan una mayor contaminación.

En este tipo de sistemas, se considera de gran importancia el cumplimiento de unos tiempos críticos, debido a la caducidad de la información obtenida y a la relevancia de las consecuencias de no cumplirse. Las consecuencias de no responder dentro de los marcos de tiempo establecidos ante un obstáculo o ante un cambio involuntario de carril pueden ser fatales. Para ello se debe utilizar hardware dedicado en vez de hardware de propósito general como puede ser un PC, estos elementos consiguen mejores resultados que los de propósito general en igualdad de características.

Actualmente existen en desarrollo algunos vehículos con estas características como pueden ser el coche de Google, siendo el más conocido mundialmente, Volvo, BMW, Mercedes… El vehículo autónomo de Google es capaz de circular por sí solo, ya se ha aprobado la circulación de estos coches de forma legal por el estado de Nevada, Estados Unidos. Para ello dispone de los dispositivos necesarios para el reconocimiento de los carriles, señales de tráfico, detección de cruces, vehículos peatones y obstáculos (Figura 5), tomando decisiones para evitar cualquier percance y llegar al destino marcado. Para ello dispone de los siguientes elementos:

LIDAR: es un dispositivo que detecta objetos y mide la distancia a ellos mediante haces de láser. Este le permite crear una recreación tridimensional del entorno.
RADAR: dispone de cuatro radares, tres de ellos en la parte delantera del coche y el cuarto en la trasera. El delantero central mide las distancias con los obstáculos que hay enfrente, los delanteros laterales supervisan los carriles adyacentes o los vehículos estacionados, mientras que el trasero hace las funciones de retrovisor y de medidor de distancias durante la circulación marcha atrás.
Cámara: para el reconocimiento de señales, semáforos y líneas de la calzada. También se puede hacer uso de cámaras para controlar el estado del conductor, analizando su conducta o su estado físico, recomendando acciones a tomar o tomando el control del vehículo si fuera necesario.
Encoder: dispone de uno en la rueda trasera para medir la distancia recorrida.
GPS: proporciona información sobre la posición del vehículo.
Unidad inercial: combinado con el GPS proporcionan una mayor exactitud en la localización del vehículo.

Figura 5: Percepción del entorno del vehículo inteligente de Google

Por otra parte, los vehículos inteligentes no solo se componen de los sensores de ayuda a la navegación, también incluyen sistemas de comunicación para poder transmitir información de forma inalámbrica entre los coches que hay en circulación, ayudando a reducir el número de accidentes y el número de congestiones del tráfico. Se estima que en Estados Unidos se pierden 3.900 millones de galones[7] de combustible solo por esta causa. Ford tiene una gran implicación en estos sistemas, equipando a los vehículos con sistemas de comunicación Wi-Fi o mediante señales de corto alcance, permitiendo obtener datos de forma omnidireccional y con visibilidad obstruida a diferencia de los sistemas de RADAR, LIDAR que son unidireccionales.

2.3.3.2. Vehículos de exploración

En este tipo de vehículos, el sistema de navegación escogido será determinante. Debido a la composición de terreno, los sistemas de odometría tradicionales por encoders no son lo suficientemente fiables para determinados terrenos, esto es debido a que puede encontrarse con superficies sobre las cuales las ruedas deslicen respecto del suelo, registrando el movimiento de la rueda pero sin haber desplazamiento sobre la superficie. Pueden existir también situaciones con un relieve rocoso, en las cuales, puede darse la posibilidad de que no todas las ruedas estén en contacto con la superficie (Figura 6). Esto supone que el vehículo no alcance la posición indicada (Figura 7), requiriendo mayores tiempos y esfuerzos en conseguir dicho objetivo.

Otro problema surge en la navegación por satélite, hay que tener en cuenta que no todas las regiones disponen de cobertura de los satélites, por lo que se necesitan de otros métodos para la localización. Estos casos podrían ser en interior de edificios, cuevas, en zonas de vegetación espesa, o en otros planetas, como es el caso del Spirit y el Curiosity en Marte.

Los rover hacen uso de odometría visual con ayuda de una cámara estéreo. Mediante un operador de detección de esquinas se seleccionan aquellos puntos con los valores de interés

[7] 1 galón corresponde a un volumen de 3,7854118 litros.

más altos. Estos puntos de cada cámara se emparejan, obteniendo la posición tridimensional de ellos. Después de que el rover se haya desplazado, se toma un segundo par de imágenes de la nueva posición y se proyectan los puntos del primer par sobre el segundo par de imágenes por medio de aproximación de movimiento basada en la rueda odométrica. Entonces una búsqueda basada en la correlación establece las posiciones bidimensionales en el segundo par de imágenes (Figura 8). El emparejamiento estéreo determina las nuevas posiciones tridimensionales. Con las dos representaciones tridimensionales se procede a estimar el desplazamiento realizado [7].

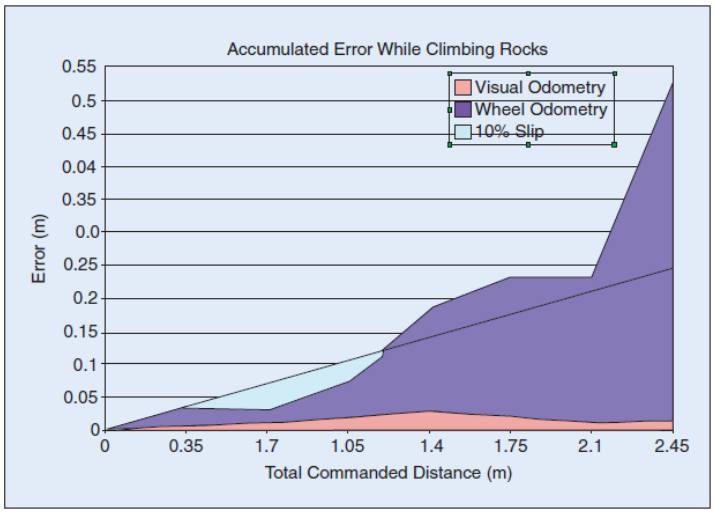

Figura 6: Errores de posición según distintos métodos navegación [7]

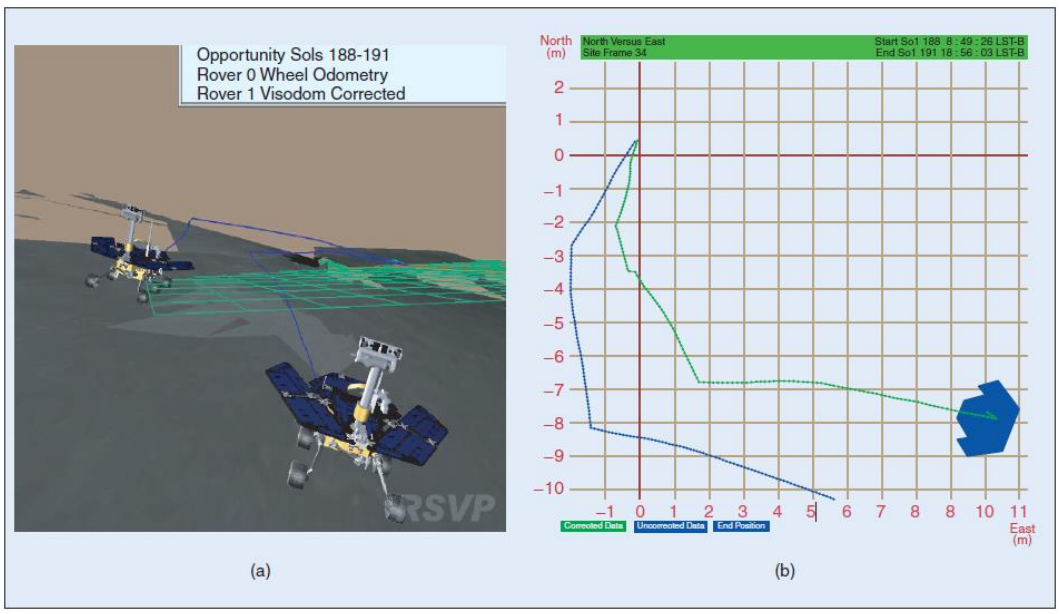

Figura 7: Reproducción en la trayectoria con odometría visual (verde) y con estimación por los sensores inerciales y los encoders (azul) [7]

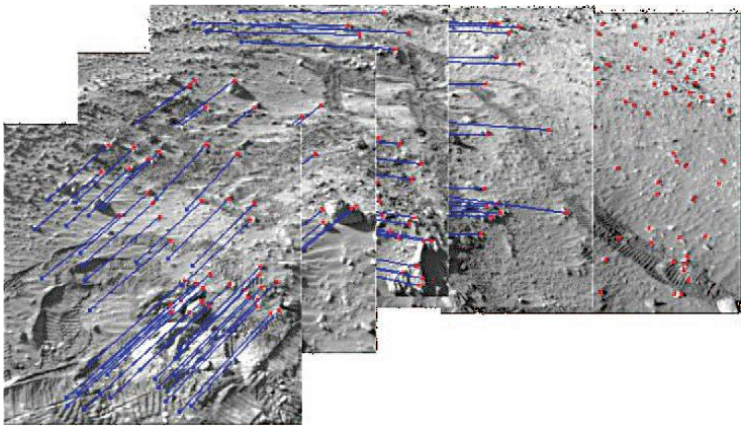

Figura 8: Etapa de emparejamiento de los puntos de interés del rover Spirit [7]

2.3.3.3. Periféricos

Los periféricos de los ordenadores como pueden ser los ratones, hacen uso de este principio para poder controlar un cursor en una interfaz gráfica por medio del movimiento de este periférico. Hay 2 clases que hacen uso de este principio: El ratón óptico y el láser. Su diferencia radica en las longitudes de onda que el emisor utiliza para fotografiar la superficie, obteniendo mayor precisión el láser al obtener un mayor número de puntos de referencia de la superficie que el óptico.

2.4. Sistemas de localización

Los sistemas de localización son aquellos que permiten obtener la posición, y en ocasiones la orientación, de un dispositivo dentro del entorno en el cual el sistema proporciona su servicio.

2.4.1. GNSS

Los sistemas de navegación por satélite son aquellos métodos de localización que hacen uso de una infraestructura formada por una constelación de satélites que orbitan el planeta. Estos sistemas se basan en el concepto de TOA (Time Of Arrival), este concepto mide el tiempo que tarda una onda en viajar desde el emisor al receptor, y conociendo la velocidad de la onda se conoce la distancia entre ambos.

Para determinar la posición exacta (asumiendo que no hay errores debido a la meteorología, entorno o desfase en la sincronización de los relojes) se necesitan mínimo 3 satélites. Con la distancia calculada a cada satélite, y la posición de los satélites son conocidas por sus efemérides, se pueden trazar esferas virtuales con origen en cada satélite y radio la distancia entre el satélite y el receptor y el punto de corte de las esferas es la ubicación.

En la realidad, debido a los errores, las esferas no cortan en un punto, sino que intersecan formando un volumen o crean una región que ningún satélite abarca. En esas regiones es donde se encuentra el receptor. Para corregir estos errores se hacen uso de estaciones en tierra o correcciones por software.

2.4.2. Localización por Wi-Fi y redes móviles

A diferencia de los sistemas GNSS, la localización por Wi-Fi y redes móviles no se basa en el concepto de TOF, sino que para hacen uso de la intensidad de la señal o de la calidad de enlace

para realizar la estimación de la localización [8]. Ambas magnitudes decrecen en relación al aumento de la distancia al origen, por lo que se puede extraer una aproximación de la posición.

- Medida de la potencia de señal:
  - Vector de potencia: se recoge la información de la potencia de las señales en una base de datos que incluyan las magnitudes de potencia y posición durante una etapa de entrenamiento, y para localizar se comparan las medidas obtenidas con la base de datos.
  - Triangulación de potencia: Se obtienen las potencias de varias redes, que se pueden transformar a distancias mediante las ecuaciones de propagación de ondas. Con las distancias se calcular fácilmente la ubicación resolviendo un problema de triangulación.
- Medida de la calidad de enlace:
  La calidad de enlace se mide en función de la cantidad de errores que se cometen en el envío de los paquetes en una transmisión, habiendo una relación directa entre el número de errores y la distancia al emisor.

2.4.3. Localización por redes móviles

El dispositivo móvil emite una señal que es detectada por las antenas que proporcionan el servicio de telefonía, obteniendo los datos de la posición en función de los valores registrados. Similar al funcionamiento del GPS.

2.4.4. Reconocimiento de patrones

Marcadores de radiofrecuencia, identificadores codificados en imágenes, vías cromáticas… Cada señal o identificador tiene asociado un código almacenado en una base de datos que le proporciona información sobre su ubicación

2.4.5. SLAM

Las siglas del término SLAM provienen de Simultaneous Localization and Mapping. Este método nace como respuesta a los métodos odométricos, ya que tienen unos errores acumulados que afectan a la precisión de la localización. El método consiste en generar un mapa del entorno y conseguir la ubicación dentro de dicho mapa mientras el robot se desplaza [9]. Para ello se necesita disponer de un mapa del entorno dentro del cual va a estar posicionado el robot. Dentro de los problemas de localización se pueden distinguir diversos casos:

- Seguimiento de la posición: En este problema, se conoce la posición inicial del robot, por lo que resulta ser el más fácil. Se soluciona realizando un cálculo odométrico comparándose las imágenes adquiridas con las de una región acotada del mapa o con una imagen previa. Este problema es el más extendido debido a que en la mayoría de aplicaciones suele ser conocida al menos la posición inicial del robot.
- Localización global: La posición inicial del robot es desconocida, por lo que en este caso, al comparar las imágenes adquiridas con las del mapa debemos asumir que los únicos limites van a ser los del tamaño del mapa, ya que cualquier posición puede ser posible. Debido a que la región de búsqueda es mayor que en el caso anterior, la dificultad también aumenta.
- Problema del robot secuestrado: El caso de mayor dificultad es aquel en el que el robot no conoce su posición inicial ni el mapa del entorno. Para resolver el problema se deberá

suponer que se conoce la posición inicial e ir recreando el entorno a medida que se desplaza a través de él. Con ello se consigue modelar el entorno y localizarse en él de forma simultánea.

Para resolver este método se pueden usar 2 técnicas de representación del entorno: un mapa de celdas ocupadas o marcas (Landmarks en la literatura anglosajona). La primera de ellas consiste en realizar una representación del entorno creando una malla de celdas binarias que indican si una coordenada está ocupada en la realidad o no. Las técnicas de localización más comunes son los filtros de partículas y los filtros de histogramas [9].

2.5. Sistemas de reconstrucción de entornos

Los sistemas de reconstrucción de entornos son aquellos sistemas que permiten recrear, mediante un modelo virtual, un escenario real. Para ello se sirven de un método de medición de distancias que calcula las coordenadas de los objetos físicos del entorno y un algoritmo para generar dicho modelo mediante la superposición de múltiples mediciones. Con este algoritmo conseguimos referenciar cada una de las nuevas mediciones tomadas desde múltiples orígenes a un único sistema de referencia Los sistemas de medición pueden ser cualquiera de los citados en el apartado de sistemas visuales de medición. Actualmente, uno de los sistemas que más poder mediático está adquiriendo es el Project Tango de Google.

Project Tango y EyesMap de ecapture son dos dispositivos similares diseñados con la finalidad de recrear los espacios u objetos de la realidad en un formato tridimensional para visualizar las diferentes perspectivas sobre una pantalla. Ambos dispositivos son capaces de realizar la opción de reconstrucción de entornos tridimensionales, aunque quizás el EyesMap de ecapture es más completo debido a que está diseñado para un fin más técnico. Dispone de la opción de medición de ángulos, longitudes, áreas y volúmenes y no solo realiza modelos de entornos, sino también de objetos tanto estáticos como dinámicos. De estos 2 dispositivos mencionados se va a escribir sobre el Project Tango de Google debido a la mayor información disponible acerca de su funcionamiento. El hardware encargado de las funciones de reconstrucción del entorno es:

- Cámara OV4682 RGB IR de 4 MPx desarrollada por OmniVision sensible a los espectros de los colores rojo, verde y azul y al espectro del infrarrojo.
- Cámara OV7251 desarrollada por OmniVision con un ángulo de visión de 180^{o} que graba en blanco y negro y es usada para detectar y medir el movimiento.
- Proyector de infrarrojos de Mantis Vision que junto con la primera cámara mide las distancias de los objetos enfocados.

Mediante software, los haces proyectados (Figura 9) sobre el sensor son transformados en coordenadas, para ello, reconoce como están distribuidos los haces formando bloques, ya que cada uno de estos bloques tiene una estructura única, y por triangulación es capaz de obtener las coordenadas del bloque, que junto con las medidas proporcionadas por el sensor inercial y las medidas de movimiento proporcionadas por la cámara, es capaz de añadir nuevos puntos al mapa con menor esfuerzo computacional [10].

Otros dispositivos que se han utilizado para este fin, aunque no fueron diseñados originalmente para ello es la Kinect de Microsoft. Con estos dispositivos, debido a que no tienen

un sensor específico para seguir el movimiento, hay que hacer una concatenación de nubes de puntos y calcular el desplazamiento entre ambas mediante funciones de error.

Existen algoritmos que hacen uso de los descriptores SIFT, calculando la distancia euclídea entre los descriptores y combinados con el algoritmo RANSAC se consiguen aquellos pares de descriptores que están espacialmente alineados. Este algoritmo está basado únicamente en los colores. Otro algoritmo desarrollado que hace uso tanto del color como de la forma es el de unión de planos. El primer paso es la extracción de los planos haciendo uso del algoritmo RANSAC, los planos seleccionados será aquellos que contengan un mayor número puntos de la muestra proporcionada. Cada plano es descrito por su histograma de color RGB y por los coeficientes de su ecuación. Y para cada par de imágenes se hacen coincidir los distintos planos encontrados.

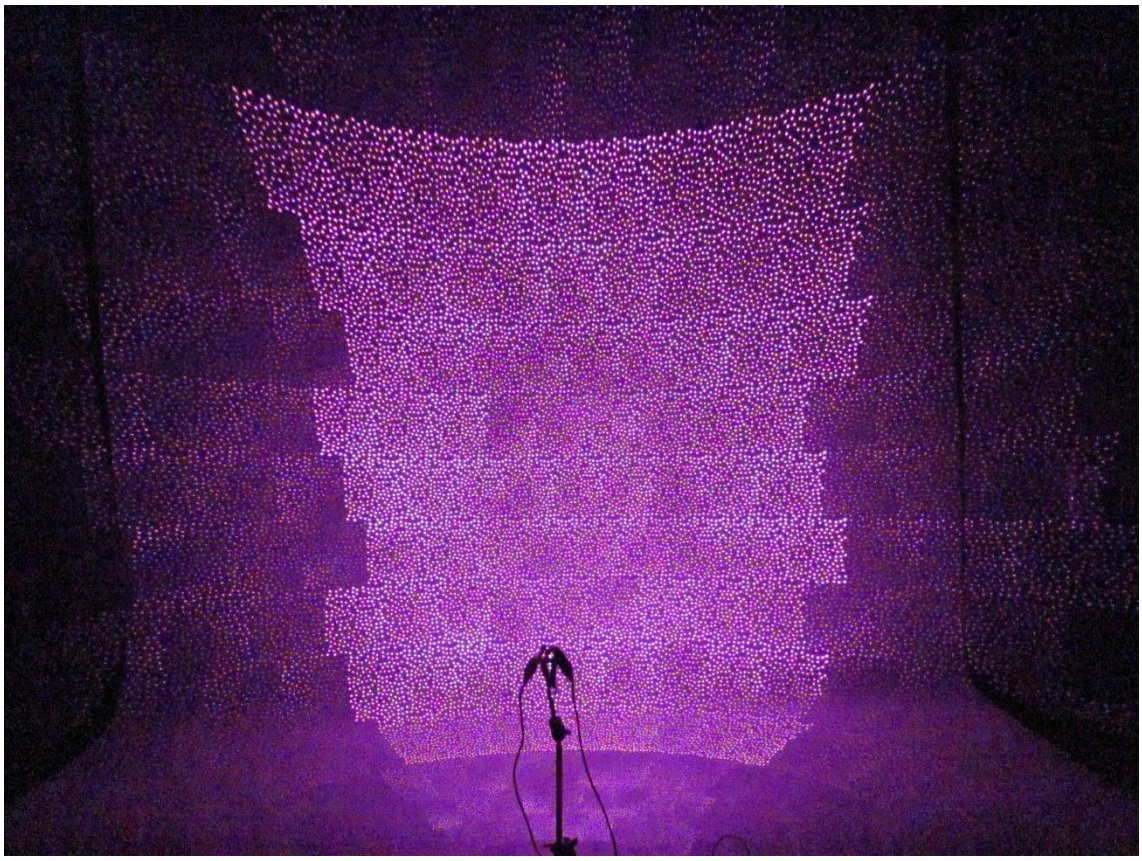

Figura 9: proyección de haces infrarrojos del Project Tango. Fuente IFIXIT

2.6. Procesamiento paralelo, CUDA

El procesamiento paralelo es la técnica utilizada para ejecutar diversas instrucciones al mismo tiempo, reduciendo así los tiempos invertidos respecto al procesamiento tradicional secuencial. Para ello, el programa ha de dividirse en diversos fragmentos de código que puedan ser ejecutados simultáneamente. Una de sus mayores ventajas que presenta este sistema es la escalabilidad, aunque la mejora de rendimiento no tiene un crecimiento proporcional al número de hilos de código que se ejecutan a la vez. Esto es debido a los problemas que surgen al acceder a recursos compartidos.

El procesamiento secuencial, a diferencia del procesamiento paralelo, es aquel en el que las instrucciones se ejecutan una vez que la instrucción anterior ha terminado. La programación estructurada asegura que mediante tres reglas es capaz de ejecutar cualquier tarea. Estas son: secuencialidad, condicionalidad, repetitividad.

- La secuencialidad es la forma de ejecutar las instrucciones una detrás de otra.
- La condicionalidad es la imposición de unas reglas que permiten acceder o bloquear la ejecución de instrucciones en función de los valores de las condiciones.
- Y la repetitividad permite la ejecución de ciertas instrucciones un determinado o indeterminado número de veces.

2.6.1. Tipos de arquitecturas

Existen diversos tipos de arquitecturas de procesamiento paralelo, cada una con sus ventajas e inconvenientes frente a las otras. En función de la tarea que se va a realizar se optará por una arquitectura o por otra.

❖ multiprocesamiento simétrico (SMP): es aquella en la que los procesadores comparten la misma memoria y los mismos buses. La existencia de una única zona de memoria simplifica los diseños de programación y de arquitectura hardware, siendo el sistema operativo quien gestiona la información de la memoria compartida entre los distintos procesadores. Este sistema tiene una desventaja, que a medida que aumenta el número de procesadores, tanto el bus como la memoria se saturan, bloqueándose unos procesos a otros y disminuyendo el rendimiento general del programa (Figura 10).

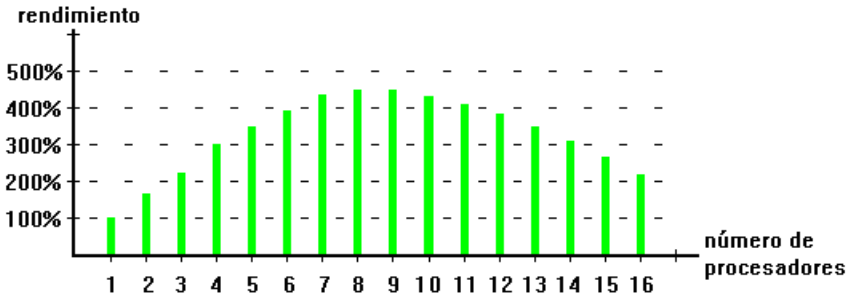

Figura 10: Medida del rendimiento de un sistema frente al número de procesadores con arquitectura SMP

❖ procesamiento masivamente paralelo (MPP): en esta arquitectura, para evitar cuellos de botella en el bus, no se hace uso de memorias compartida. En su lugar se hace un reparto de la memoria entre los distintos procesadores, y para que un procesador pueda acceder a datos de una memoria ajena a la suya se hace uso de un esquema de paso de mensajes análogo a los paquetes de datos en redes, consiguiendo así una reducción del tráfico del bus con el resultado de un mejor aprovechamiento de los recursos hardware (Figura 11). El inconveniente es la complejidad de programación debido a la distribución de las memorias de cada procesador. Además, el intercambio de datos tiene mayor complejidad que haciendo uso de una memoria global compartida. Este tipo de soluciones tienen un mayor costo por procesador, por lo que no es adecuado para cualquier actividad.

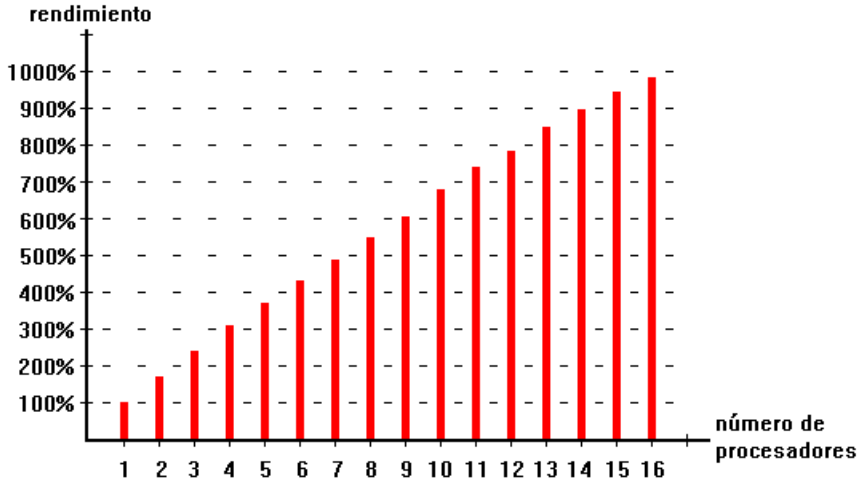

Figura 11: Medida del rendimiento de un sistema frente al número de procesadores con arquitectura MPP

- procesamiento paralelo escalable (SPP): es una combinación de los dos sistemas anteriores. Hace uso de una memoria jerárquica de dos niveles. El primer nivel es aquel que hace uso de memorias distribuidas al igual que en el MPP, y el segundo nivel de memoria esta globalmente compartida como en el SMP. Con esta nueva arquitectura se facilita la programación respecto a los sistemas MPP y aumenta la escalabilidad respecto a los SMP (Figura 12).

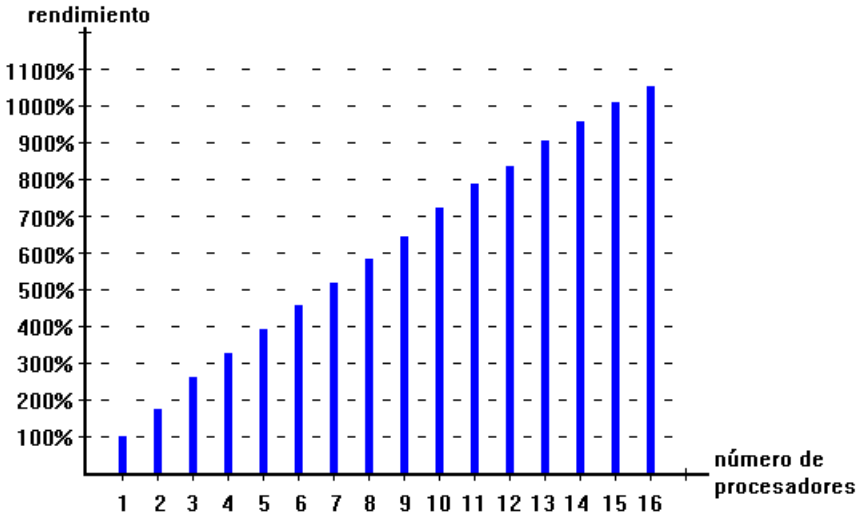

Figura 12: Medida del rendimiento de un sistema frente al número de procesadores con arquitectura SPP

La técnica habitual para aumentar la potencia de los sistemas ha sido mediante el aumento del número de núcleos por procesador, el de procesadores lógicos que operan en un equipo o aumentando el número de equipos montando clusters de ordenadores (Figura 13). En este último caso, las soluciones habituales para servidores han sido servidores montados en racks o servidores blade.

Figura 13: Clúster de ordenadores de la NASA compuesto por 20 equipos Altix con 10240 CPUs

Figura 14: Servidor montado en rack

Los rack son soportes destinados a albergar múltiples equipos electrónicos o informáticos (Figura 14). Con esta disposición se optimiza el espacio utilizado por equipo, mientras que los servidores blade son una clase de ordenador diseñado para optimizar el espacio y el consumo (Figura 15). Éstos a diferencia de los ordenadores montados en rack no son completamente funcionales, ya que no disponen de todos los elementos de un ordenador. Esto se hace como medida para minimizar el espacio ocupado.

Figura 15: Servidor blade de IBM

2.6.2. Calculo acelerado en la GPU

Se puede definir como el uso combinado de una unidad de procesamiento gráfico y una CPU para acelerar el procesamiento de las aplicaciones. Para ello se hace que aquellas partes de código que requieran mayor carga computacional y que puedan obtener una mejora de rendimiento al paralelizarse se ejecuten en la GPU, dejando el resto para su ejecución en la CPU.

A esta técnica de programación se le denomina GPGPU (General Purpose computing on Graphics Processing Units). En esta técnica se hace uso de la GPU para aprovechar sus capacidades de cálculo, ya que estas han sido especialmente diseñadas para operaciones en coma flotante, además de la mejora de rendimiento (potencia de cálculo por vatio de potencia consumida) de éstas respecto a las CPU.

El uso de tarjetas gráficas en la computación de propósito general puede tener su origen en el año 2002, cuando ATI Technologies saco al mercado su Radeon 9700. Una unidad de procesamiento visual (VPU), cuyos núcleos adquirían una flexibilidad similar a los de las CPUs y Nvidia lanzó sus unidades de procesamiento de flujo genérico GeForce 8. De esta forma, las GPUs pasan a ser un elemento de computación más general que lo que había hasta la fecha, dando paso a lo que hoy se conoce como GPGPU. Debido al desarrollo de estas capacidades, surgió el estándar OpenCL, que permite desarrollar código para CPUs y GPUs centrándose especialmente en la portabilidad. Sin embargo, Nvidia desarrollo su propio lenguaje exclusivo para sus GPUs denominado CUDA, destinado también al ámbito de la computación sobre GPUs. Las tarjetas gráficas han sido utilizadas desde entonces para realizar múltiples aplicaciones en diversos campos como son:

Dinámica molecular
Química cuántica
Tecnología de materiales
Software de visualización
Bioinformática
Análisis numérico
Física
Defensa e inteligencia
Finanzas
Dinámica de fluidos
Análisis estructural
Diseño asistido
Edición de imágenes y video
Codificación
Previsión meteorológica
Medicina…

Según los mecanismos de control se pueden clasificar en tres tipos:

1. Single Instruction Single Data (SISD): los equipos con este mecanismo cuentan con una unidad de control, un procesador y una memoria, y existe un único flujo de instrucciones sobre un único flujo de datos.
2. Single Instruction Multiple Data (SIMD): una única unidad de control maneja el flujo de instrucciones que opera sobre múltiples datos. La unidad de control gobierna múltiples procesos.

Una variante a esta técnica creada por Nvidia se denomina Single Instruction Multiple Thread (SIMT), la diferencia reside en que en esta última técnica, cada dato es operado en un hilo diferente, mientras que en la técnica de SIMD, los múltiples datos son procesados en un mismo procesador simultáneamente.

3. Multiple Instruction Single Data (MISD): distintas instrucciones operan sobre el mismo conjunto de datos. Pueden darse dos situaciones para este caso:
   - Varias instrucciones operando sobre un único dato
   - Varias instrucciones operando sobre un dato que se modifica sirviendo de entrada para la siguiente etapa.
4. Multiple Instruction Multiple Data (MIMD): tienen múltiples flujos de instrucciones que operan sobre múltiples datos. Capa proceso puede operar de manera independiente.

Las GPUs están formadas por Streaming Multiprocessors (SM), con múltiples núcleos por cada uno de ellos. Por ello las aplicaciones diseñadas para ejecutarse en las GPUs deben ser capaces de lanzar múltiples hilos a la vez que puedan ejecutarse en paralelo para así explotar al máximo las capacidades de las GPUs. Aquellos hilo pertenecientes a una misma trama se asignan a un SM y se ejecutan en un SIMD (Single Instruction Multiple Data). La ventaja que presentan los SIMD es que todos los núcleos ejecutan la misma instrucción al mismo tiempo, por lo que solo se decodifica una vez la instrucción.

En el siguiente caso, se hace una comparación entre la potencia de cálculo de la GPU frente a la de la CPU en el análisis de imágenes médicas por computación.

Para una media de 600 pacientes que son tratados en el hospital anualmente y 5-6 muestras de imágenes de alta resolución por cada paciente, la aplicación que funciona con Matlab tarda 21 meses en procesar toda la información en un PC. Mientras que con una GPU el tiempo de procesado disminuye a los 2.4 días utilizando CUDA.

Para hacerse una idea del uso de las GPUs en la computación, dentro del ranking de las computadoras más potentes del mundo se pueden encontrar varias haciendo uso de GPUs, como el caso de la Cray Titan, actualmente en segunda posición tras la supercomputadora Tianhe-2, y que antes ocupó la primera posición. En la siguiente tabla se muestran las prestaciones de distintos modelos de procesadores de propósito general frente a procesadores gráficos. Confirmando que ofrecen mayor rendimiento en aplicaciones de procesado masivo de datos (Tabla 1).

Tabla 1: Comparativa de unidades de procesamiento tipo CPU y GPU

	intel xeon e5 2687w	intel xeon e7 8890 v2	intel xeon phi 7120	tesla k40	Titan Z
# of Cores	8	15	61	2880	5760 (2x2880)
# of Threads	16	30			
Clock Speed	3.1 GHz	2,8 GHz	1,238 GHz	745 MHz	705 MHz
Max Turbo Frequency	3.8 GHz	3,4 GHz	1,333 GHz	810-875 MHz	876 MHz
Scalability	2S	8S			
Max TDP	150 W	155 W	300 W	235-245 W	375 W
Max Memory Size (dependent on memory type)	256 GB	1536 GB	16	12	12
Memory Types	DDR3-800/1066/1333/1600	DDR3-800/1066/1333/1600	GDDR5	GDDR5	GDDR5
# of Memory Channels	4	4	16		
Max Memory Bandwidth	51.2 GB/s	85 GB/s	352 GB/s	288 GB/s	672 GB/s
Memory Interfaces				384 bit	768 bit (2x384)
Base FLOPs Single Core	198,4 GFLOPs				
Single Core Max FLOPs	230 GFLOPs		2 TFLOPs	4,29 TFLOPs	8 TFLOPs
Base FLOPs (Double Precision)			1 TFLOP	1,43 TFLPOs	

2.6.3. CUDA

CUDA (Compute Unified Device Architecture) es una arquitectura de cálculo paralelo desarrollada por NVIDIA dedicada a aprovechar al máximo el potencial de sus GPUs, aumentando así el rendimiento del sistema. CUDA proporciona varias extensiones de C y C++ para implementar el paralelismo de las tareas en función de la granularidad[8] deseada. Para realizar un programa en CUDA se deben conocer algunos conceptos que ayuden a entender su funcionamiento:

- **Streaming processor**: cada uno de los procesadores individuales que son capaces de ejecutar instrucciones.
- **Streaming multiprocessor**: a nivel de hardware, conjunto de streaming processors agrupados bajo el mismo control y bajo una misma memoria compartida.

Al llamar al kernel de la función de CUDA, se invocan 'N' bloques con 'M' **threads** cada uno. Estos threads o hilos se ejecutan en paralelo en los Streaming processors.

- Los **bloques** son el conjunto de threads que ejecutan el kernel, pudiendo comunicarse entre sí los threads de cada bloque a través de la memoria compartida, mientras que los threads de distintos bloques no.
- Al conjunto de bloques se les denomina **grid o malla**.
- Los threads de cada bloque se agrupan en grupos de un tamaño establecido en función de la arquitectura de la tarjeta gráfica formando **warps**, estos son las unidades más pequeñas que se ejecutan simultáneamente en cada Streaming multiprocessor. El resto de warps permanecen a la espera de que termine el que está activo o realizan tareas de

[8] Granularidad: se puede definir como la relación entre el tiempo en realizar una operación de comunicación básica y el tiempo requerido por un cómputo básico.

búsqueda de datos para agilizar la ejecución. Todos los threads activos de un warp ejecutan la misma instrucción, y estos pueden activarse o desactivarse en función de las ramificaciones que sufra el kernel. Por ejemplo, las instrucciones IF, WHILE, FOR… son instrucciones que aumentan el nivel de ramificación del programa, esto implica que ciertos threads realicen una función diferente a los otros, aumentando el tiempo de ejecución.

Las tarjetas gráficas de Nvidia hacen uso de una arquitectura SPP, con una jerarquía en los tipos de memorias. Cada uno de los núcleos dispone de una memoria local exclusiva, donde los demás núcleos no tienen acceso. Para compartir datos con otros núcleos del mismo bloque hacen uso de la memoria compartida, y para comunicar núcleos de distintos bloques disponen de la memoria global.

Debido a que CUDA es un lenguaje de programación exclusivo para un tipo de tarjetas gráficas, y que éstas pueden ser diseñadas en función de dicho lenguaje de programación, el conjunto de tarjeta gráfica Nvidia y CUDA produce una sinergia positiva mejorando su rendimiento frente a otros lenguajes multiplataforma como OpenCL. Hay ciertas diferencias entre OpenCL y CUDA que hacen declinarse por una opción u otra. OpenCL es un lenguaje diseñado para que pueda ser portado entre diferentes dispositivos: CPUs, GPUs… Para ello, la compilación se realiza en tiempo de ejecución, por lo que esos tiempos se sumaran a los tiempos de ejecución. Para respaldar la valoración dada se ofrecen datos sobre comparaciones de ambos lenguajes [11].

Para medir el rendimiento se ha hecho uso de la aplicación Adiabatic QUantum Algorithms (AQUA), es una simulación Monte Carlo de un sistema de spin cuántico escrito en C++. En él se aproxima la configuración del spin cuántico con el clásico sistema del spin Ising. La aproximación clásica consiste en copias ferro-magnéticamente acopladas del sistema cuántico. Cada copia esta acoplada con otras dos copias, formando un anillo. A este proceso de aproximación se le denomina descomposición Suzuki-Trotter. Los experimentos que han realizado abarcan el rango desde los 8 qubits (bits cuánticos) hasta los 128, y el número de capas es 128 para todo el experimento (Tabla 2).

Tabla 2: Tamaño de los sistemas cuánticos y sus correspondientes tamaños clásicos [11]

Qubits	Layers	Simulation Points	Classical Spin Variables
8	128	27	27,648
16	128	34	69,632
32	128	37	151,552
48	128	57	350,208
72	128	71	654,336
96	128	111	1,363,968
128	128	129	2,113,536

Cada problema del experimento se ha realizado un total de diez veces para tener un valor estadísticamente relevante de dicho experimento. Los resultados sobre los tiempos

transcurridos durante el desarrollo de dicha aplicación se muestran a continuación (Tabla 3 y Tabla 4). En ellos se muestran los valores de la media y la desviación estándar, la media calcula el tiempo promedio que tarda en solucionarse el problema, y la desviación estándar mide la incertidumbre de los valores frente a la media.

Tabla 3: Tiempos de ejecución y de aplicación [11]

Qubits	GPU Operations Time				End-To-End Running Time			
	CUDA		OpenCL		CUDA		OpenCL	
	avg	stdev	avg	stdev	avg	stdev	avg	stdev
8	1.97	0.030	2.24	0.006	2.94	0.007	4.28	0.164
16	3.87	0.006	4.75	0.012	5.39	0.008	7.45	0.023
32	7.71	0.007	9.05	0.012	10.16	0.009	12.84	0.006
48	13.75	0.015	19.89	0.010	17.75	0.013	26.69	0.016
72	26.04	0.034	42.32	0.085	32.77	0.025	54.85	0.103
96	61.32	0.065	72.29	0.062	76.24	0.033	92.97	0.064
128	101.07	0.523	113.95	0.758	123.54	1.091	142.92	1.080

Tabla 4: Tiempos de ejecución del kernel y de transferencia de datos [11]

Qubits	Kernel Running Time				Data Transfer Time			
	CUDA		OpenCL		CUDA		OpenCL	
	avg	stdev	avg	stdev	avg	stdev	avg	stdev
8	1.96	0.027	2.23	0.004	0.009	0.007	0.011	0.007
16	3.85	0.006	4.73	0.013	0.015	0.001	0.023	0.008
32	7.65	0.007	9.01	0.012	0.025	0.010	0.039	0.010
48	13.68	0.015	19.80	0.007	0.061	0.010	0.086	0.008
72	25.94	0.036	42.17	0.085	0.106	0.006	0.146	0.010
96	61.10	0.065	71.99	0.055	0.215	0.009	0.294	0.011
128	100.76	0.527	113.54	0.761	0.306	0.010	0.417	0.007

Las conclusiones de dicho experimento indican CUDA ofrece un mejor rendimiento en cuanto a la transferencia de datos desde y hasta la GPU, y en cuanto a la ejecución del kernel, CUDA fue más rápido también. Para poder extraer la conclusión de que CUDA ofrece estas ventajas frente a OpenCL remarcan que las implementaciones de ambas aplicaciones son parecidas.

3. ARQUITECTURA GLOBAL

En este capítulo se desarrolla el esquema del método realizado, comentando cada uno de los pasos y de los diferentes procesos que se llevan a cabo. Primero, se procede con una breve explicación de los distintos pasos que componen el método, y posteriormente se profundizará en cada uno de los pasos para explicar el funcionamiento del método.

3.1. Esquema general de control

Para el desarrollo de este método se ha optado por dividir el método en 2 pasos:

- Un primer paso cuya función es la de modelar el entorno
- El segundo es la de localización de un mapa de profundidad muestra en el modelo creado en la función anterior.

En ambos pasos se pueden distinguir 3 diferentes bloques. Las distintas funciones pertenecerán a los diferentes bloques en función del objetivo que busquen alcanzar. Las acciones de cada bloque no son idénticas para ambos casos, ya que cada uno realiza una función diferente dentro del método. Los bloques en los que se ha dividido cada uno de los pasos se muestran en la Figura 16.

Figura 16: Diagrama general

A continuación se expone una breve descripción del comportamiento general de cada uno de los bloques.

En el paso de reconstrucción del entorno:

- Adquisición: El programa adquiere la información del par de cámaras y del sensor inercial. Por medio de las imágenes se genera el mapa de profundidad, necesario para los siguientes bloques.
- Búsqueda: Para detectar aquellos elementos comunes que nos servirán posteriormente para generar el mapa, hacemos uso de los algoritmos escogidos con la ayuda de un sensor inercial para reducir el área de búsqueda.
- Acción: Se añaden al mapa aquellos elementos que aportan información nueva al mapa en la ubicación proporcionada por el algoritmo.

En paso de localización los bloques son los siguientes:

- Adquisición: El programa adquiere la información del par de cámaras generando el mapa de profundidad.
- Búsqueda: Se buscan aquellas regiones del mapa que tengan una mayor correlación con la muestra generada.
- Acción: Se selecciona aquella región del mapa que contenga una probabilidad mayor de coincidir con la muestra y se proporciona la información de la manera escogida.

Para optimizar los tiempos de ejecución se ha optado por dividir cada uno de los pasos en diferentes procesos. De esta forma, las tareas a realizar pueden ser ejecutadas en paralelo. Los procesos están agrupados de la siguiente forma. En el primer paso del método se generan 3 procesos diferentes: el Proceso Padre y el Proceso Hijo 2 que son los encargados del bloque de Adquisición, mientras que el Proceso Hijo 1 de los bloques de Búsqueda y Acción. En el segundo paso del método son generados 2 procesos: el Proceso Padre realiza el bloque de Adquisición mientras que el Proceso Hijo 3 de los bloques de Búsqueda y Acción. A continuación se explica detalladamente las tareas que realizan cada uno de ellos.

3.2. Proceso Padre

El proceso padre es el núcleo del método. Alrededor de este proceso giran los demás procesos. Este proceso es el encargado de realizar las operaciones de calibrado de cámaras, adquisición de imágenes, y de realizar el mapa de profundidad que será utilizado en cualquiera de los 2 pasos del método. Además es el encargado de gestionar los recursos compartidos para evitar conflictos entre los distintos procesos al acceder a ellos y de crear los distintos procesos en función del paso del método que se encuentre realizando en dicho momento.

El funcionamiento del proceso es el siguiente (Figura 17): el proceso comienza realizando una petición para calibrar las cámaras o en su defecto leer los parámetros intrínsecos de ellas de un fichero donde se encuentran almacenados de un proceso de calibración anterior. Una vez ya cargados los valores se procede a la inicialización de los distintos elementos que van a operar: se crean zonas de memoria compartida para compartir datos entre los diferentes procesos, semáforos para sincronizar los distintos procesos y se configuran los detalles de la cámara como su resolución y tasa de refresco. Se le realizará una petición al usuario para ejecutar el paso 1, el de modelado del entorno, o el paso 2, localización, y en función de la respuesta obtenida se creará el proceso hijo correspondiente a dicho paso.

Si la opción escogida es la de modelado del entorno, el programa entra en un bucle donde permanecerá indefinidamente hasta que el usuario decida finalizarlo, dando por terminado el modelado del entorno. En dicho bucle, se procede a la captura de ambas imágenes, con ambas imágenes ya cargadas se procede a calcular su disparidad y con ella su mapa de profundidad. Esta va a ser la información que es enviada al proceso anteriormente creado, Proceso Hijo 1. Para ello, antes de almacenar los datos en la memoria compartida, se comprueba que ningún proceso esté realizando alguna operación sobre dicha memoria compartida. En el momento que la memoria compartida está disponible, se procede a bloquear el acceso de otros procesos a esa zona de memoria, para evitar así que se puedan leer datos mientras se están escribiendo nuevos. Una vez terminado de escribir se permitirá de nuevo el acceso liberando el semáforo y vuelve a iniciarse el bucle. Cuando el usuario envía la orden de que el modelo se ha finalizado, el Proceso Padre se mantiene a la espera de que los 2 procesos creados hayan concluido. Dando por finalizada el primer paso del método.

Por contrario si la opción escogida es la de localización, se procede a crear el proceso correspondiente a la localización, Proceso Hijo 3, y se ejecuta una única vez los pasos de adquisición de las imágenes, cálculo del mapa de disparidad y del mapa de profundidad. Dicha información es enviada al Proceso Hijo 2 de idéntica forma que con la opción anterior. Una vez finalizada la transmisión del mapa de profundidad, el Proceso Padre permanece a la espera de que el Proceso Hijo 2 finalice. Dando por finalizado el segundo paso del método.

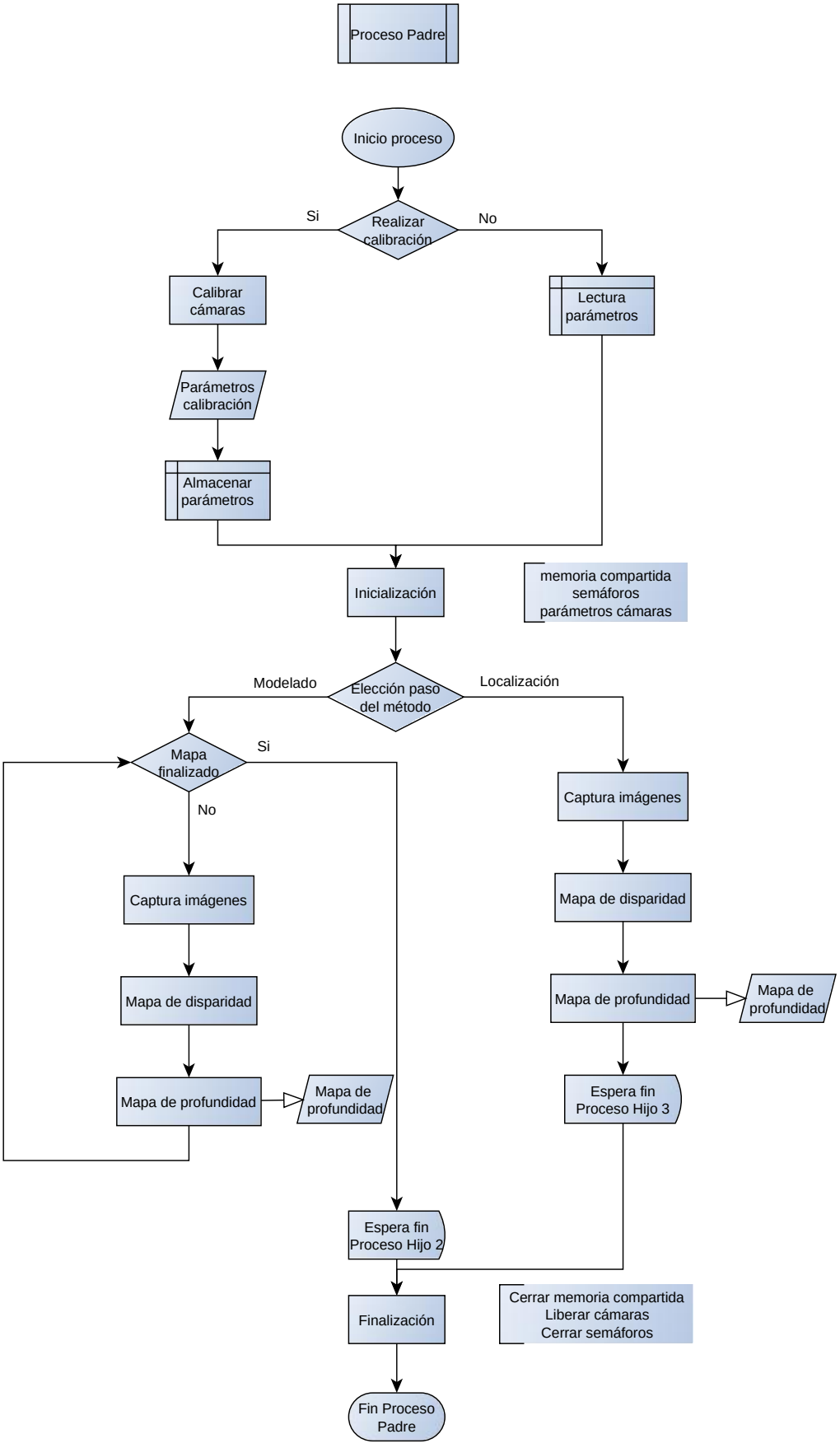

Figura 17: Diagrama de flujo del Proceso Padre

3.3. Proceso Hijo 1

El proceso Hijo 1 es el proceso encargado de realizar el modelado del entorno. Para ello debe encontrar aquellas regiones de solapamiento entre los mapas de profundidad recibidos y el modelo del entorno que él mismo genera. Para obtener los datos necesarios para el algoritmo de alineamiento de nubes de puntos es necesario crear un proceso encargado de proporcionar unas medidas aproximadas del movimiento.

Las etapas de este proceso son las siguientes (Figura 18): en el apartado de inicialización se procede a abrir los recursos creados por el programa padre como son la memoria compartida (Proceso Padre-Proceso Hijo 1), que a partir de ahora se llamará memoria compartida 1, a abrir los semáforos y crea una nueva zona de memoria compartida (Proceso Hijo 1- Proceso Hijo 2), que se llamara memoria compartida 2. En la siguiente etapa, de ejecución, el proceso permanece en un bucle hasta que el usuario decida finalizarlo. En este punto, el proceso se queda a la espera de poder leer los datos de la memoria compartida 1, correspondiente al mapa de profundidad. Al acceder a ella bloquea su acceso y al finalizar su lectura la vuelve a liberar para una nueva escritura por parte del Proceso Padre. Estos pasos se realizan de igual forma para leer los datos de la memoria compartida 2, leyendo la matriz de transformación homogénea que se ha calculado en el Proceso Hijo 2.

Del mapa de profundidad leído se extrae una muestra aleatoria, se le aplica la matriz de transformación a dicha muestra para realizar un cambio de coordenadas, de un sistema de coordenadas local, fijo a las cámaras, a un sistema de coordenadas absoluto fijado en la posición de la primera captura de las imágenes. Con el cambio de coordenadas de la muestra se realiza una búsqueda de dicha muestra dentro del mapa que se ha generado hasta el momento. Se corrigen aquellos errores en la matriz de transformación y se añaden los elementos nuevos al mapa, ampliándose en cada iteración del bucle.

3.4. Proceso Hijo 2

El proceso Hijo 2 es el encargado de realizar las lecturas del sensor inercial, y de calcular la matriz de transformación homogénea correspondiente a la posición y orientación actual. El proceso únicamente establece comunicación con la plataforma arduino y lee los datos procedentes de un sensor inercial por el puerto serie. Para ello se debe realizar la inicialización de las comunicaciones estableciendo los parámetros necesarios como el identificador del puerto y la velocidad de la comunicación. Con esas lecturas calcula la matriz de transformación asociada desplazamiento realizado en cada intervalo temporal en base a los desplazamientos realizados. Para tener una mejor aproximación de la matriz de transformación homogénea, se debe tener en cuenta las correcciones que realiza el Proceso Hijo 1 sobre ella. Por ello se realiza una lectura de la matriz de transformación homogénea actualizada por el Proceso Hijo 1 de la memoria compartida 2 y combina ambas matrices de transformación para obtener la posición actual, enviándose de nuevo al Proceso Hijo 1 por medio de la memoria compartida 2 (Figura 19).

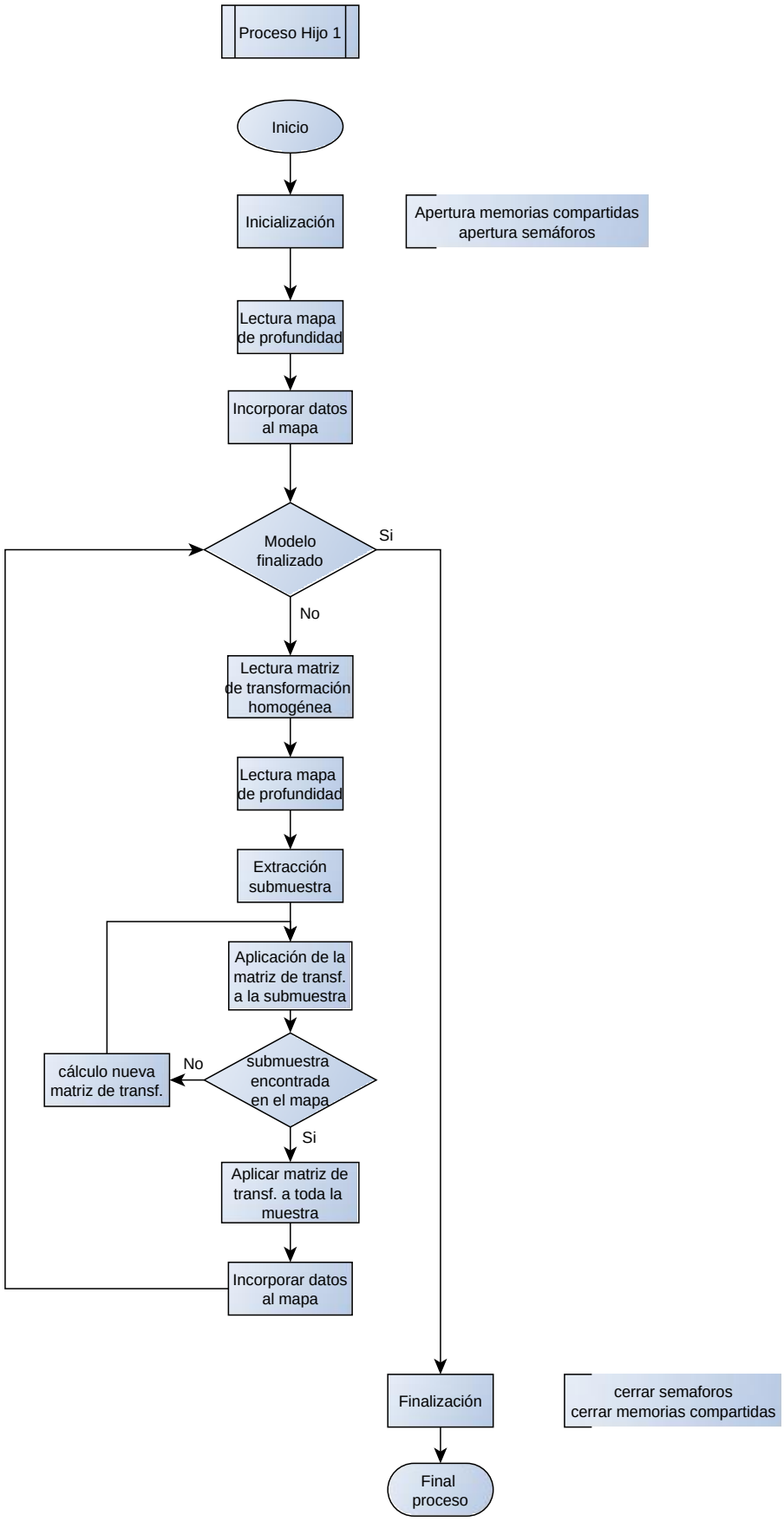

Figura 18: Diagrama de flujo del Proceso Hijo 1

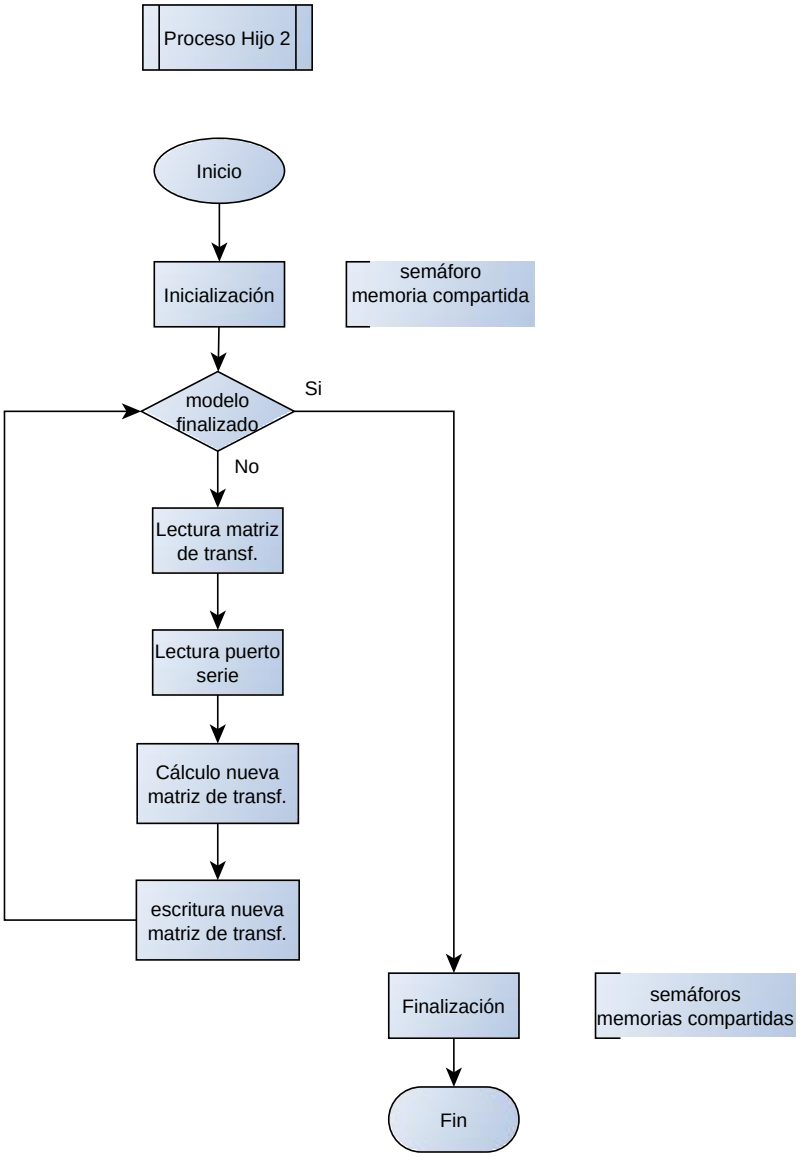

Figura 19: Diagrama de flujo del Proceso Hijo 2

3.5. Proceso Hijo 3

En este proceso ocurre el paso de localización. Se realiza una modelo tridimensional del modelo del mapa y del mapa de profundidad para realizar una búsqueda exhaustiva de una región del mapa de profundidad sobre el modelo del mapa.

El funcionamiento de este proceso es el siguiente (Figura 20): el proceso realiza una lectura del modelo del entorno y del mapa de profundidad, éste último proviene del Proceso Padre, y para cada uno de ellos y en base a sus dimensiones crea una malla de celdas con un tamaño de celda preestablecido en función de la resolución deseada. Con ambas mallas ya creadas, se procede a realizar una correlación de una región de la malla creada a raíz de las imágenes del robot con la malla del modelo. Esta correlación, a diferencia de las correlaciones realizadas en una imagen tradicional, se realizará en 3 dimensiones y en distintos planos por las posibles, siendo en total 6 ejes, aunque para algunas aplicaciones, el número de ejes será menor debido a que algunos grados de libertad pueden estar bloqueados por la propia estructura del

robot Con este procedimiento se obtienen unos valores que nos permitirán decidir entre las distintas ubicaciones posibles.

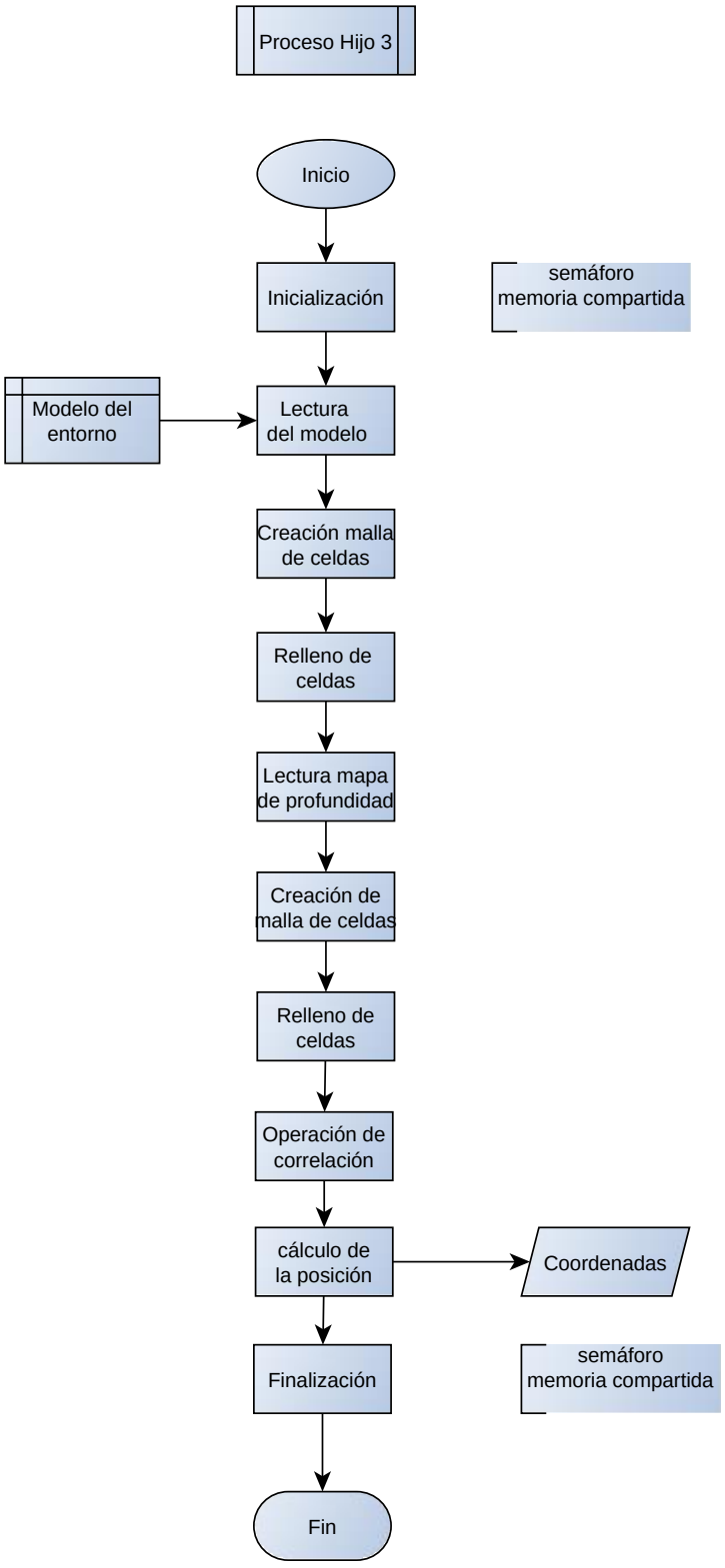

Figura 20: Diagrama de flujo del Proceso Hijo 3

4. IMPLEMENTACIÓN

En este apartado se van a explicar los conceptos que se han aplicado durante el desarrollo de este método, incorporando las ecuaciones matemáticas que rigen el comportamiento de las distintas funciones realizadas para crear un modelo de ellas. Las distintas funciones y sus principios son los siguientes:

4.1. Visión

Para dotar de las capacidades visuales al método se han realizado las siguientes funciones: primeramente se han de inicializar las cámaras para obtener las imágenes con las características deseadas. Para el paso de proyectar las imágenes bidimensionales sobre el espacio tridimensional debemos realizar una etapa de calibración. Con ella conseguimos reducir el nivel de las aberraciones geométricas que incorpora la cámara. Esta corrección se le aplicará a cada una de las imágenes capturadas, realizando además una operación de filtrado para disminuir los niveles de ruido que puedan existir. Una vez terminada esta etapa de pre procesamiento, se procede a calcular el mapa de disparidad y con éste, el mapa de profundidad. Terminando así las funciones de visión.

Las funciones de visión y de tratamiento de imágenes se han realizado a través de las librerías de OpenCV. Han sido las cuales han transformado las imágenes capturadas por las cámaras en la información útil necesaria en el programa. Las principales funciones utilizadas en el programa son:

- **Adquisición de imágenes:**

Se especifican los parámetros de la imagen como son la resolución, la velocidad de captura o los colores. Además de corregir la distorsión mediante los parámetros de calibración de las cámaras.

La calibración es el proceso mediante el cual se consiguen los parámetros de las cámaras que permiten calcular las coordenadas de un objeto dentro de la imagen en base a las coordenadas que ocupa en la realidad. Se necesita calibrar las cámaras porque debido a las aberraciones geométricas de la óptica y del sensor, se proyectan los objetos en una posición distinta a la que deberían ocupar [12].

Modelo lente fina:

En este modelo, se hacen converger los rayos que inciden de forma paralela al eje óptico mediante el uso de una lente (Figura 21) según las siguientes ecuaciones:

$$\frac{1}{A} + \frac{1}{A'} = \frac{1}{F} \quad (1)$$

$$B' = \frac{B}{A} * A' \quad (2)$$

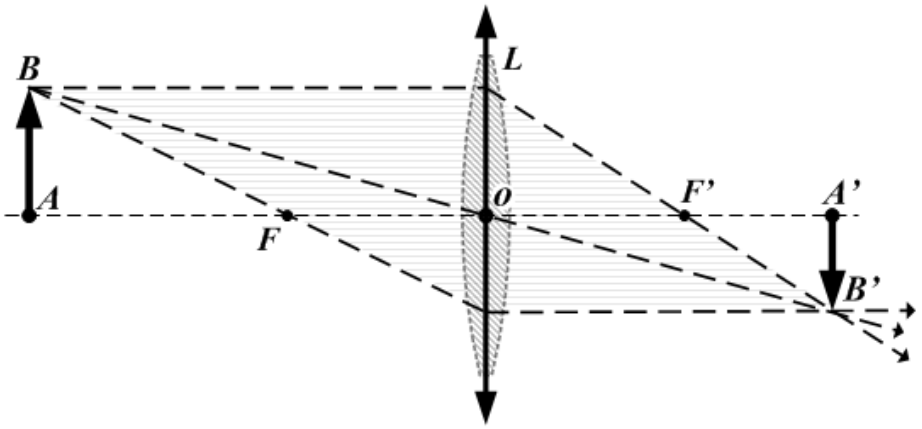

Figura 21: Diagrama del modelo de lente fina

Parámetros intrínsecos:

Son aquellos que definen la cámara, como su óptica y geometría. Definen el proceso de transformación que sufren los haces luminosos desde que atraviesan la óptica hasta que inciden sobre el elemento sensor.

- Enfoque
- Distancia focal
- Factor de escala
- Aberraciones…

Parámetros extrínsecos:

Definen la situación de la cámara dentro de un sistema de referencia global:

- Rotación
- Translación

Para hacer una estimación más exacta de la posición real de los objetos en función de su representación hacemos uso de la matriz de calibración M. Para ello es necesario hacer el cálculo de los parámetros intrínsecos [2, 12].Esto se debe a que los objetos no tienden a proyectarse en el sensor como son en la realidad, sino que sufren transformaciones transformando la geometría de la realidad.

Modelo Pin Hole:

En este modelo, el orificio de entrada debe ser lo suficientemente pequeño como para evitar que las haces de luz incidan en el sensor con varias direcciones, ya que eso provocaría una imagen borrosa. En el modelo ideal, en el sensor solo inciden los rayos que proceden de una única dirección. La dirección es aquella recta que proviene del objeto y atraviesa el orificio (Figura 22), y la posición en el sensor de cualquier punto dentro del campo de visión es la siguiente:

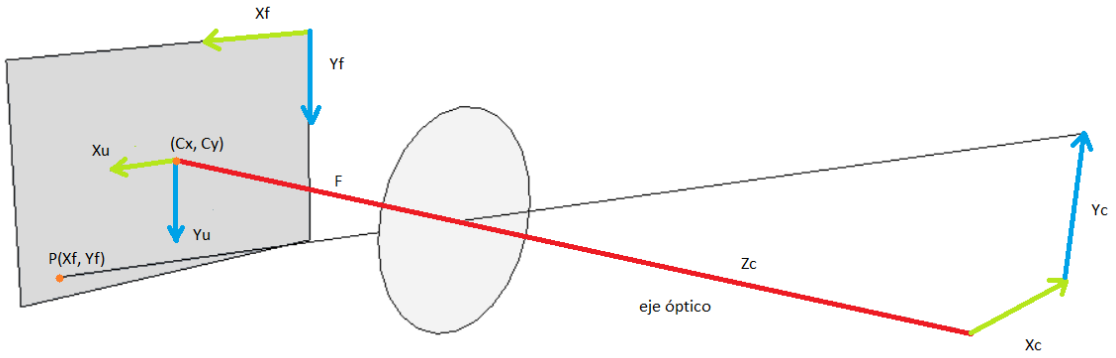

Figura 22: Diagrama del modelo Pin Hole

$$x_u = \frac{x_c}{Z_c} * F \tag{3}$$

$$y_u = \frac{y_c}{Z_c} * F \tag{4}$$

Nomenclatura:

- Coordenadas del mundo: (X_w, Y_w, Z_w).
- Coordenadas laterales de la imagen: (X_f, Y_f).
- Coordenadas centrales de la imagen: (C_x, C_y; X_u, Y_u).
- x_u , y_u son las coordenadas reales distorsionadas.
- x_d, y_d son las coordenadas teoricas sin distorsionar.
- K_1 y K_2 son los coeficientes de distorsión.
- K_x y K_y son los factores de escala.
- F es la distancia focal.
- C_x y C_y son los valores del centro de la imagen.
- r_{ij} son los valores pertenecientes a la matriz de rotación.
- t_i pertenecen a la matriz de translación.
- R es la distancia al centro de la imagen

Distancia focal

$$\frac{x_u}{F} = \frac{x_w}{z_w} \tag{5}$$

$$\frac{y_u}{F} = \frac{y_w}{z_w} \tag{6}$$

$$\begin{bmatrix} n * x_f \\ n * y_f \\ n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/F & 0 \end{bmatrix} * \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{7}$$

Coeficientes de distorsión radial

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} * (1 + K_1 * r^2 + K_2 * r^4) \quad (8)$$

Coeficientes de distorsión tangencial

$$x_u = 2 * p_1 * x_d * y_d + p_2 * (r^2 + 2 * x_d^2) \quad (9)$$

$$y_u = 2 * p_2 * x_d * y_d + p_1 * (r^2 + 2 * x_d^2) \quad (10)$$

Combinando la distorsión radial con la distorsión tangencial obtenemos la siguiente expresión:

$$x_u = x_d * (1 + K_1 * r^2 + K_2 * r^4) + 2 * p_1 * x_d * y_d + p_2 * (r^2 + 2 * x_d^2) \quad (11)$$

$$y_u = y_d * (1 + K_1 * r^2 + K_2 * r^4) + 2 * p_2 * x_d * y_d + p_1 * (r^2 + 2 * x_d^2) \quad (12)$$

Relación entre las coordenadas laterales y centrales

$$x_f = K_x * x_d + C_x \quad (13)$$

$$y_f = K_y * y_d + C_y \quad (14)$$

Matriz de proyección entre las coordenadas del mundo y las coordenadas del sensor (sin distorsión)

$$\begin{bmatrix} n * x_f \\ n * y_f \\ n \end{bmatrix} = M * \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad (15)$$

$$\begin{bmatrix} n * x_f \\ n * y_f \\ n \end{bmatrix} = \begin{bmatrix} K_x & 0 & C_x/F & 0 \\ 0 & K_y & C_y/F & 0 \\ 0 & 0 & 1/F & 0 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad (16)$$

Modelo Pin Hole vs lente fina:

Pin Hole

$$y_u = \frac{y_c}{z_c} * F \quad (17)$$

Lente fina

$$B' = \frac{B}{A} * A' \quad (18)$$

$$-\frac{B}{A} * A' = B - \frac{B}{F} * A' \quad \rightarrow \quad A' = \frac{A * F}{A - F} \tag{19}$$

$$B' = \frac{B}{A} * \frac{A * F}{A - F} \quad \rightarrow \quad B' = \frac{B * F}{A - F} \tag{20}$$

Tabla 5: Equivalencias entra las nomenclaturas de ambos modelos

Pin Hole	Lente fina
z_c	A
y_c	B
F	F
y_u	B’
/	A’

$$\frac{y_u}{B'} = \frac{\frac{y_c}{z_c} * F}{\frac{B * F}{A - F}} \quad \rightarrow \quad \frac{y_u}{B'} = \frac{A - F}{Zc} \tag{21}$$

$$F \ll A \quad \rightarrow \quad \frac{y_u}{B'} \simeq \frac{A}{Zc} = 1 \tag{22}$$

Como se puede observar, ambos modelos se pueden tomar por semejantes haciendo una conversión en las nomenclaturas según la

Tabla 5 y una pequeña aproximación numérica, haciendo válidas las ecuaciones de las distorsiones y las ecuaciones de proyección del modelo Pin-Hole al de lente fina.

Mapa de disparidad.

Debido a la separación relativa entre las 2 cámaras, se pueden observar diferencias entre las imágenes tomadas por cada una de ellas. Según Marr y Poggio, son tres las etapas necesarias para calcular la disparidad entre dos imágenes:

1. Seleccionar los puntos característicos de una imagen.
2. Encontrar los puntos los mismos puntos característicos de la primera imagen en la segunda.
3. Medir la distancia relativa entre las posiciones de los puntos característicos de ambas imágenes.

La función de la biblioteca OpenCV encargada de calcular la disparidad entre las 2 imágenes hace uso del algoritmo Block Matching (BM). Éste es un método utilizado para la detección de movimiento basado en la eliminación de la redundancia entre dos o más fotogramas [13]. Es un método comúnmente utilizado en algunos métodos de compresión de video.

Para ello divide la imagen actual es bloques cuadrados de tamaño N que no se solapen. Y para cada bloque de referencia de la imagen actual se busca el bloque que mejor coincida en una ventana cuadrada del tamaño 2*W+N en la imagen anterior, siendo W el máximo desplazamiento permitido. Quedando la posición relativa entre ambos bloques representada por un vector.

Para seleccionar el bloque que se adapte mejor al bloque (x,y) seleccionado se usa la función de error $D_p(i,j)$.

$$D_p(i,j) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} |f_t(x,y) - f_t(x+i, y+j)|^p \quad (23)$$

Si p=1, el criterio de comparación se denomina: suma de diferencias absolutas (SAD).

Si p=2, se denomina suma de diferencias cuadradas (SSD).

i,j son las coordenadas del vector desplazamiento que se prueba en cada iteración. Los diferentes métodos difieren en como seleccionar estos valores: Full Search (FS), Three Steps Search (3SS) [14]…

- **Mapa de profundidad**

El objetivo final de obtener el mapa de disparidad es el de poder medir distancias. Para ello se hace uso de la triangulación. La triangulación es el cálculo de posiciones o medidas desconocidas mediante el uso de ecuaciones trigonométricas. Como se verá a continuación, la distancia del objeto enfocado está relacionada con la disparidad que produce en el par de cámaras. Para hacer un modelo de mayor simplicidad el par de cámaras estará formado por 2 elementos iguales en cuanto al sensor, a la óptica y a su construcción.

Como se puede comprobar, el triángulo con vértices P-O_i-O_d es semejante al triángulo formado por la unión de los triángulos P_i-C_i-O_i y P_d-C_d-O_d.(Figura 23 y Figura 24). Por lo que podemos aplicar el teorema de Tales para encontrar las coordenadas del punto P.

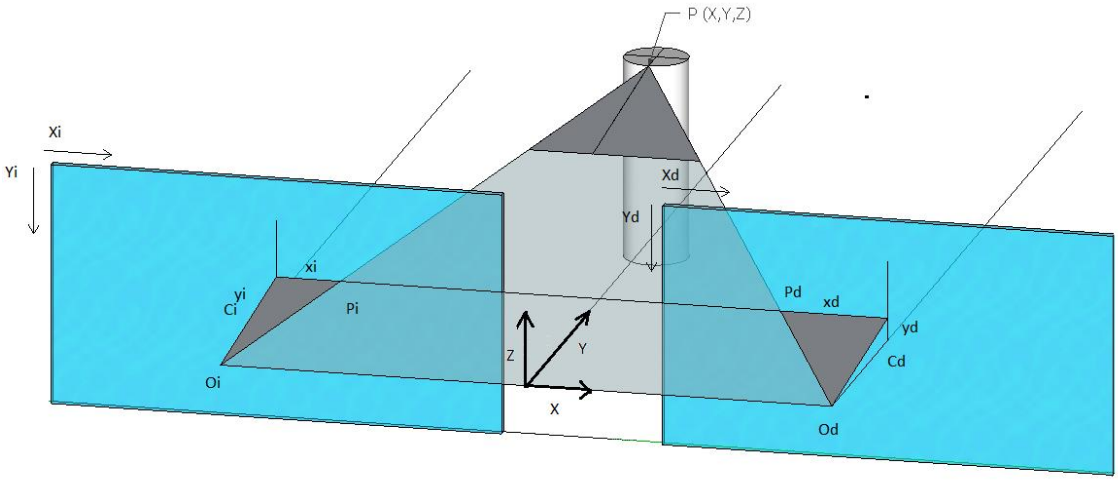

Figura 23: Trazado de triángulos semejantes en las proyecciones de un punto

Se determina el plano formado por los segmente $\overline{P\ O_i}$, $\overline{P\ O_d}$ y $\overline{O_i O_d}$. Los segmentos $\overline{P\ O_i}$ y $\overline{P\ O_d}$ intersecan en los puntos $P_i(x_i, y_i)$ y $P_d(x_d, y_d)$ respectivamente. Proyectando sobre el plano Y conseguimos la siguiente disposición:

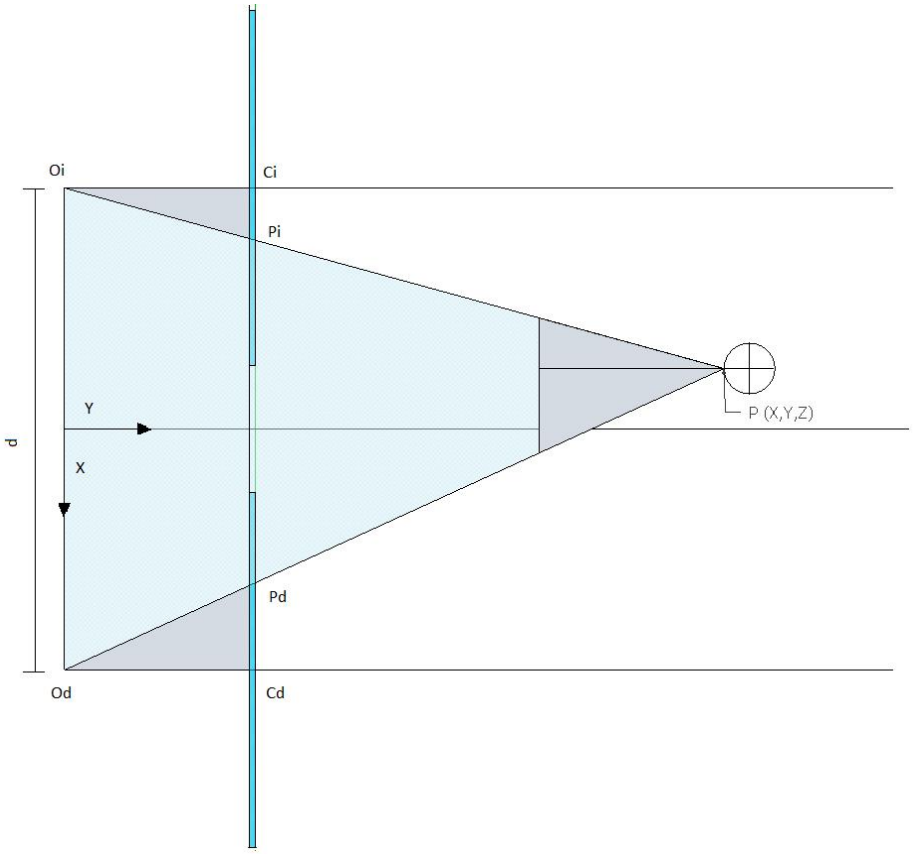

Figura 24: Planta de la representación de las proyecciones de un punto

$$\frac{Y}{\overline{O_l O_d}} = \frac{F}{\overline{C_l P_l(x)} + \overline{C_d P_d(x)}} \tag{24}$$

La disparidad D tiene la siguiente expresión:

$$D = P_i(x) - P_d(x) \tag{25}$$

$$P_i(x) = C_i + x_i \tag{26}$$

$$P_d(x) = C_d - x_d \tag{27}$$

Haciendo los cambios de variable:

$$D = C_i + x_i - C_d + x_d = x_i + x_d \tag{28}$$

$$\overline{C_l P_l(x)} = x_i \tag{29}$$

$$\overline{C_d P_d(x)} = x_d \tag{30}$$

$$\overline{O_l O_d} = d \tag{31}$$

$$\frac{Y}{d} = \frac{F}{x_i + x_d} = \frac{F}{D} \tag{32}$$

Para la coordenada Z;

$$\frac{y_d}{F} = \frac{Z}{Y} \tag{33}$$

Y para la coordenada X:

$$\frac{\frac{d}{2} - X}{Y} = \frac{x_d}{F} \tag{34}$$

4.2. Odometría

Para determinar con mayor precisión y rapidez la posición correcta del par de cámaras dentro del espacio se ha optado por combinar los datos proporcionados por la estimación odométrica y por la odometría visual: La estimación odométrica acota la región de búsqueda para el algoritmo ICP, reduciendo el número de iteraciones a realizar y la odometría visual reduce los errores proporcionados por los sensores inerciales.

4.2.1. Navegación inercial

Para determinar la posición del par de cámaras durante la recreación del entorno se hace uso de 2 técnicas diferentes: odometría inercial y odometría visual. Para medir el movimiento se ha hecho uso de los siguientes sensores inerciales: un acelerómetro de 3 ejes y un giróscopo de 3 ejes. Las magnitudes obtenidas de cada uno de ellos son una aceleración lineal (a_i) expresada en $\left[\frac{m}{s^2}\right]$ para el acelerómetro y una velocidad angular (ω_i) expresada en $\left[\frac{rad}{s}\right]$ para el giróscopo. Ambas magnitudes deben ser integradas en el tiempo para obtener unas magnitudes espaciales $(\Delta S_i), (\Delta\theta_i)$.

$$\Delta S_i(t) = \int_0^{\Delta t} v_i(t) * dt \tag{35}$$

$$v_i(t) = \int_0^{\Delta t} a_i(t) * dt + v(t-1) \tag{36}$$

$$\Delta S_i(t) = \frac{1}{2} * a_i(t) * \Delta t^2 + v(t-1) * \Delta t \tag{37}$$

$$\Delta\theta_i(t) = \int_0^{\Delta t} \omega_i(t) * dt = \omega_i(t) * \Delta t \tag{38}$$

Calculados los desplazamientos en un intervalo temporal, hay que añadir estos nuevos movimientos a los ya realizados anteriormente. Para ello será necesario hacer un cambio de ejes, ya que los desplazamientos anteriormente descritos están referenciados a los ejes de los sensores, y no a los ejes absolutos de nuestro sistema de referencia fijo [15]. La forma utilizada para representar el movimiento es mediante las matrices de transformación homogéneas (T_i):

$$MTH_i = \begin{bmatrix} R_{3x3_i} & T_{3x1_i} \\ f_{1x3_i} & E_{1x1_i} \end{bmatrix} = \begin{bmatrix} Matriz\ de\ rotación & Vector\ de\ translación \\ Vector\ de\ perspectiva & Magnitud\ de\ escalado \end{bmatrix} \tag{39}$$

(40)

$$R_{3x3} = \mathrm{R_{Yaw}} * \mathrm{R_{Roll}} * \mathrm{R_{Pitch}}$$

R_{Yaw} R_{Roll} R_{Pitch}

$$\begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}\quad\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi & \cos\phi\end{bmatrix}\quad\begin{bmatrix}\cos\theta & 0 & -\sin\theta\\ 0 & 1 & 0\\ \sin\theta & 0 & \cos\theta\end{bmatrix} \tag{41}$$

$$R=\begin{bmatrix}c\psi c\theta & c\psi s\theta s\Phi - s\psi c\Phi & c\psi s\theta c\Phi + s\psi s\Phi\\ s\psi c\theta & s\psi s\theta s\Phi + c\psi c\Phi & s\psi s\theta c\Phi - s\psi s\Phi\\ -s\theta & c\theta s\Phi & c\theta s\Phi\end{bmatrix} \tag{42}$$

R_{Roll} = rotación sobre eje X

R_{Pitch} = rotación sobre eje Y

R_{Yaw} = rotación sobre eje Z

$$T_{3x1}=\begin{bmatrix}T_x\\ T_y\\ T_z\end{bmatrix} \tag{43}$$

Debido a que los ejes del sensor están alineados con los ejes del móvil (strap-down) y no con unos ejes absolutos (gimbaled) hay que modificar los valores de salida del sensor para adaptarlos a las ecuaciones anteriores. La salida del acelerómetro en la posición de reposo es la siguiente:

$$\begin{bmatrix}a_x^0\\ a_y^0\\ a_z^0\end{bmatrix}=\begin{bmatrix}0\\ 0\\ g\end{bmatrix} \tag{44}$$

Si los ejes del robot difieren de los ejes absolutos, entonces el término de la gravedad se verá reflejado en los demás términos de la aceleración.

$$\begin{bmatrix}a_x^0\\ a_y^0\\ a_z^0\end{bmatrix}=\begin{bmatrix}g_x\\ g_y\\ g_z\end{bmatrix} \tag{45}$$

Para calcular los valores de la aceleración sobre los ejes absolutos, se deben proyectar las aceleraciones medidas por los sensores sobre ellos. Convirtiendo así las magnitudes desde el sistema de referencia móvil al absoluto. Los valores a_x, a_y, a_z son los que nos interesan para introducir en las ecuaciones para obtener los desplazamientos, los cuales deberán ser tratados para cambiar de un sistema de referencia fijo al robot por uno absoluto.

$$a_X = a^0 * \begin{bmatrix}1\\ 0\\ 0\end{bmatrix} * \vec{\imath} \tag{46}$$

$$a_Y = a^0 * \begin{bmatrix}0\\1\\0\end{bmatrix} * \vec{j} \tag{47}$$

$$a_Z = a^0 * \begin{bmatrix}0\\0\\1\end{bmatrix} * \vec{k} \tag{48}$$

Una vez proyectadas las aceleraciones sobre el sistema de coordenadas absoluto se elimina el término asociado a la gravedad del eje Z, por lo que las medidas resultantes pertenecen únicamente a las de la aceleración del robot.

El vector de perspectiva f_{1x3} va a ser el vector nulo debido a que estamos trabajando con distancias y no con imágenes.

$$f_{1x3} = [0 \quad 0 \quad 0] \tag{49}$$

Y por último la magnitud de escalado será la unidad debido a que no se harán modificaciones en las ópticas una vez iniciado el proceso.

$$E_{1x1} = 1 \tag{50}$$

Para conocer la posición respecto al origen de coordenadas absoluto de las coordenadas obtenidas por el par de cámaras en el instante **J**, debemos hacer las siguientes operaciones:

$$\begin{bmatrix}x\\y\\z\end{bmatrix} = MTH_0^1 * MTH_1^2 * \ldots * MTH_{J-1}^J * \begin{bmatrix}x_j\\y_j\\z_j\end{bmatrix} \tag{51}$$

4.2.2. Odometría visual

La odometría visual es un método de localización basado en la obtención de información relativa a los desplazamientos mediante el análisis de imágenes. Para ello se hace uso de algoritmos que seleccionan los puntos invariantes de una imagen: son aquellos puntos que son menos propensos a sufrir variaciones debido a la iluminación o a transformaciones geométricas. Algunos de los detectores de puntos característicos más utilizados son el SIFT (**Scale-invariant feature transform**) o el SURF (**Speeded Up Robust Features**), que está basado en el método anterior.

Con el mapa de profundidad ya obtenido no vamos a requerir de utilizar ningún detector de puntos característicos, ya que cada uno de los puntos del mapa con una asignación en el espacio es útil para calcular los desplazamientos mediante el algoritmo ICP[9].

[9] El algoritmo ICP está destinado a alinear dos nubes de puntos diferentes pero pertenecientes a una misma realidad.

Algoritmo ICP

El algoritmo ICP (Iterative Closest Point) es un método por el cual se pretenden alinear 2 nubes de puntos diferentes con elementos comunes entre ellas [16]. Hay diversas técnicas para conseguir dicho alineamiento que pueden agruparse bajo 2 grupos: las técnicas basadas en el algoritmo de Chen y Medioni y las basadas en el de Besl y Mckay. En el último de ellos, es necesario que una muestra sea parte de la otra y se debe conocer una aproximación inicial.

Para ello se parte de una nube de puntos M, que será la muestra con la cual queramos referenciar el resto de puntos, y una muestra D, que será la que queramos ubicar en la posición correspondiente de M. El algoritmo consiste en minimizar el error entre los puntos de la nube de puntos S con sus correspondientes de la muestra M según la siguiente ecuación.

$$e = \sum_{i=1}^{N}\sum_{j=1}^{M} \omega_{ij} * \left\| M_i - (R * S_j + t) \right\|^2 \tag{52}$$

Donde:

M_i son los puntos del pertenecientes a la recreación del entorno.
S_j son los puntos pertenecientes a la muestra tomada.
ω_{ij} es la matriz que indica si existe correspondencia entre los puntos M_i y D .
R es la matriz de rotación.
t es el vector de translación.

Descripción del proceso:

1. En el caso que se va a describir se obtiene una primera aproximación de las matrices R y t por medio de un sensor inercial o grupo de ellos capaces de proveer cada una de las magnitudes.
2. Se realiza una conversión de sistemas de referencias para poder referenciar los puntos de D desde el sistema de referencia de M. La muestra D se denomina ahora D'
3. Se busca aquellos puntos de M que tengan la menor distancia con los puntos de D'. Estos puntos seleccionados de M los denominamos M_s.
4. Para empezar a resolver la ecuación, se procede a calcular el centroide de las muestras D' y de M_s, aplicando nuevamente una translación a D' igual a la diferencia de ambos centroides, obteniendo D''. La nueva función objetivo a minimizar es la siguiente:

$$e = \sum_{i=1}^{N}\sum_{j=1}^{M} \omega_{ij} * \left\| M_i - (R * D_j)' \right\|^2 \tag{53}$$

5. Para calcular la nueva matriz de rotación realizamos la descomposición de valores singulares de la matriz H.

$$H = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix} \tag{54}$$

$$S_{ij} = \sum M_i * D_j \tag{55}$$

$$H = U * \Sigma * V^t \quad (56)$$

Σ es la matriz diagonal que contiene los autovalores de H.

V es una matriz ortogonal compuesta por los autovectores de $H^t * H$

U es una matriz de columnas ortogonales en las que sus columnas son

$$u_i = \frac{A * v_i}{\sigma_i} \quad (57)$$

Siendo la matriz de rotación R calculada de la siguiente manera: $R = V * U^t$

6. El siguiente paso es calcular la matriz de translación t.

$$t = cdg_m - R * cdg_d \quad (58)$$

cdg_m es el centroide de la muestra M_s.
cdg_d es el centroide de la muestra D''.

7. Se calcula el error cometido entre la muestra D'' y M y se repiten los pasos desde el punto 3 en adelante mientras que el error disminuya o hasta que tenga una tolerancia permitida.
8. Una vez obtenida la matriz de rotación y el vector de translación que minimiza el error entre la muestra M' y D'', se aplican dichas transformaciones al total de la muestra D y se añaden a M, ampliando el número de puntos que comprenden la muestra.

4.3. Localización

Las funciones de localización se encuentran en el segundo paso del método. Para ello es necesario que se haya realizado previamente el primer paso para tener el modelo que va a servir como referencia.

Para desarrollar este método se ha optado por la operación de correlación entre el modelo y una región del mapa de profundidad, ambos convertidos en una malla de celdas. Se ha elegido dicho método por ser inmune a transformaciones geométricas, ya que las medidas de los objetos son invariantes ante dichos cambios y frente a otros como puede ser la iluminación. La forma de escoger la región del mapa de profundidades la siguiente: se eligen localizaciones aleatorias, se analizan dichas muestras y solo son seleccionadas aquellas que cumplan unas características impuestas. En este caso la condición trata de imponer un umbral de ocupación, esto significa que el número de celdas ocupadas dentro de un volumen preestablecido, con origen en los distintos puntos aleatorios, debe superar un determinado valor para considerarse como candidato. Una vez seleccionada la región, modelo B, se procede a realizar la correlación de ésta con la malla del modelo A. Para ello se divide el modelo A en distintos planos según el valor de su coordenada Z, realizando l correlación para los distintos planos X-Y. En cada iteración se obtendrá un valor en función del grado de similitud encontrado, Dicho valor de correlación servirá para estimar la opción más probable del emplazamiento del robot. Una vez que la correlación ha finalizado para estos ejes, se realiza una rotación de ellos dentro del modelo A. De esta forma abarcamos las distintas orientaciones con las que las

imágenes han podido ser tomadas las imágenes, Como se ha explicado en el capítulo 3, según la arquitectura del robot se podría realizar una correlación en 6 ejes o menos. En este caso se ha optado por una correlación de 4 ejes para simplificar el método. En el caso que el robot disponga de una brújula de 3 ejes, el proceso de correlación se transformaría automáticamente en un problema de 3 ejes.

La ecuación de la correlación para una arquitectura de 4 grados de libertad es:

$$g(x,y,z,\theta) = f(i,j,k)\ o\ g(x+i,y+j,z+k,\theta+l) = \sum_{i=0}^{N1}\sum_{j=0}^{N2}\sum_{k=0}^{N3}\sum_{l=0}^{N4} f(i,j,k)\ o\ g(x+i,y+j,z+k,\theta+l) \quad (59)$$

Para medir el índice de correlación de cada posición, se tiene en cuenta si las celdas ocupadas del modelo B coinciden con las del modelo A. Debido a la existencia de errores en las mediciones se puede tener en cuenta el valor de las celdas vecinas para eliminar en parte la influencia de ellos en la función de localización, aunque el valor que aporte el emparejamiento celda-vecino no es igual al de un emparejamiento celda-celda. Para lo que resta de documento, los únicos emparejamientos que se tomarán como válidos serán los emparejamientos celda-celda.

Una vez que se ha completado el proceso de correlación, se escogen aquellas coordenadas que maximizan el valor de la función de correlación, y con las con las coordenadas de la región escogida del mapa de profundidad, se puede calcular la posición desde la cual fueron tomadas las imágenes.

4.4. CUDA

Se han desarrollado diferentes funciones del que se ejecuten en una unidad de procesamiento paralelo. Las funciones implementadas bajo esta técnica son:

4.4.1. Multiplicación de matrices

Como se ha indicado en uno de los puntos anteriores, uno de los pasos es transformar el mapa de profundidad de un sistema de coordenadas local a uno global, para ello se ha de realizar el producto de una matriz 4x4 por múltiples vectores 4x1.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \quad B = \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \\ b_{4,1} \end{bmatrix} \quad C = \begin{bmatrix} c_{1,1} \\ c_{2,1} \\ c_{3,1} \\ c_{4,1} \end{bmatrix} \quad (60)$$

$$C = A * B \quad (61)$$

Como se puede ver, para realizar el producto de dichas dos matrices se han requerido 16 multiplicaciones y 12 sumas. Al realizar la operación de ambas matrices en serie, el número de pasos temporales serán 28, mientras que si se realizan en paralelo podemos disminuir el número de pasos temporales a 3 haciendo uso de 16 núcleos de procesamiento en la primera etapa (Figura 25).

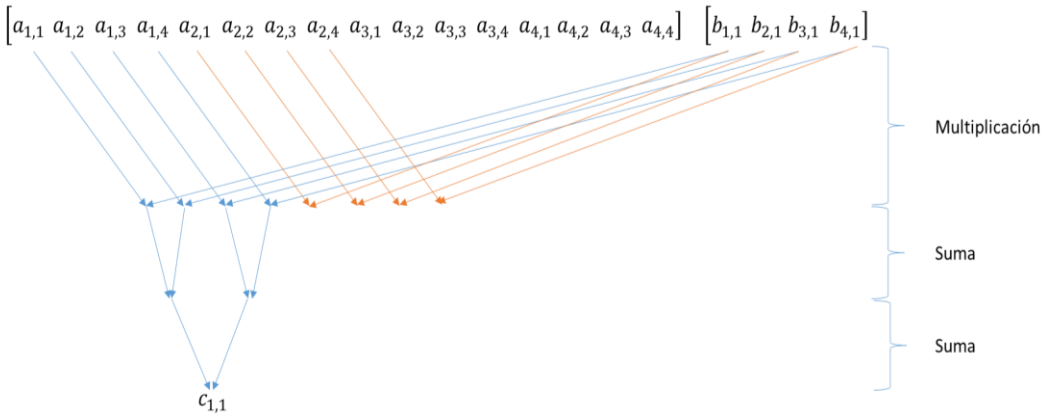

Figura 25: Esquema del producto de vectores en paralelo

Debido a que es necesario hacer el producto de múltiples vectores, se hace uso de un mayor número de núcleos de procesamiento. En la función desarrollada se ha hecho uso de una muestra de 384 vectores en una GPU de 384 núcleos de procesamiento. Debido a que el número de núcleos de procesamiento es inferior al requerido para implementar este método se ha optado por una solución hibrida entre los 2 procesos anteriormente descritos. Cada núcleo de procesamiento realiza el cálculo de 4 elementos de los nuevos vectores. Para ello realiza 4 iteraciones realizando las operaciones correspondientes con las filas y columnas asociadas a cada núcleo. Cada nuevo vector es calculado por 4 núcleos diferentes, aunque debido al tamaño de la muestra, cada núcleo calcula 4 elementos, por lo que al final es similar a que cada núcleo calcula un nuevo vector en serie, el motivo de que se haga de esta manera y no que un solo núcleo calcule un vector entero es debido a la escalabilidad. Aumentar o disminuir el tamaño de la muestra solo nos va a influir en el número de iteraciones que realicemos, mientras que de la otra forma, podemos desperdiciar recursos hardware si elegimos una muestra de menor tamaño que el número de núcleos de nuestra GPU. Por otra parte, estamos aprovechando mejor los recursos hardware de la GPU que si realizamos todo el proceso en paralelo al no disponer de los suficientes núcleos de procesamiento, ya que el número de ellos que se utilizan en cada paso se reduce a razón de 2, dejando un mayor número de ellos inactivos. La contrapartida es que el número de pasos se ve aumentado.

4.4.2. Calculo de raíces

Para realizar el cálculo de los autovalores de H, se procede a multiplicar $H^t * H$, obligando que los autovalores ahora pertenezcan al conjunto de los reales. Para calcular los autovalores podemos proceder de diversas maneras, haciendo uso del algoritmo QR realizando sucesivamente la factorización QR de la matriz H hasta que se tenga una matriz diagonal por bloques con los autovalores en la diagonal. Este proceso es muy costoso computacionalmente hablando, teniendo la factorización QR una complejidad de $O(n^3)$ y el algoritmo completo una complejidad de $O(n^4)$.

Debido a esto, procedemos con otros métodos de búsqueda de los autovalores como es el Teorema de Gerschgorin. Según dicho teorema, los autovalores de la matriz se encuentran en aquellas regiones del plano complejo delimitadas por la circunferencia de centro $C_i = H_{ii}$ y radio $r_i = \sum_{j \neq i}^{n} |H_{i\,j}|$. Una vez conseguida una región delimitada se procede a dividir la región en múltiples intervalos y a evaluar el polinomio característico en ellos haciendo uso de la GPU. Una vez obtenidos los valores, tomamos el teorema de Bolzano como primer criterio para

encontrar las raíces: se observan los cambios de signo en intervalos consecutivos (o si el punto evaluado tiene valor 0), será en esos intervalos donde se encuentre un autovalor. Un segundo criterio para determinar la posición de una raíz es la del cambio de pendiente. Debido a que se ha forzado a que las raíces sean reales, si observamos que la pendiente de la función en un punto se aproxima al eje de abscisas y en otro punto el signo de la pendiente es diferente, entonces hay una raíz entre esos 2 puntos. Esta condición se ha impuesto por la posibilidad de que haya 2 raíces muy próximas y con el primer criterio no sean detectadas debido al tamaño de los intervalos. Si se necesita una mayor resolución en el cálculo de los autovalores se actualiza el entorno de búsqueda a dicho intervalo y se procede de la misma manera: se divide, se evalúa y se observan cambios de signo, así sucesivamente. En el caso de que no se encontraran ningún cambio de signo, se generan unos intervalos de menor aumentando el número de ellos.

Una vez obtenidos los autovalores de $H^t * H$, no hay mas que realizar la raíz cuadrada sobre ellos para obtener los autovalores de H.

Ejemplo:

$$H = \begin{pmatrix} 0{,}07142 & 5{,}91858 & 0{,}18932 \\ 0{,}16426 & 11{,}882339 & 0{,}40096 \\ 0{,}11002 & 8{,}58922 & 0{,}28112 \end{pmatrix}$$

Multiplicamos la matriz por su transpuesta para que las raíces sean reales:

$$H^T * H = \begin{pmatrix} 0{,}044187 & 3{,}319484 & 0{,}110312 \\ 3{,}319484 & 249{,}994263 & 8{,}299448 \\ 0{,}110312 & 8{,}299448 & 0{,}275639 \end{pmatrix}$$

Obtenemos el polinomio característico:

$$y = -x^3 + 250{,}3140869141 * x^2 - 0{,}0548496248 * x + 0{,}0000002384$$

Las regiones delimitadas donde se encuentran los autovalores de $H^T * H$ son los círculos con centro C_i y radio r_i:

$C_1 = 0{,}044187$ $r_1 = 3{,}429796$

$C_2 = 249{,}994263$ $r_2 = 11{,}618932$

$C_3 = 0{,}275639$ $r_3 = 8{,}40976$

Los autovalores son los siguientes:

$\lambda_1 = 250{,}313858032$ (Figura 26)

$\lambda_2 = 0{,}000214687$ (Figura 27)

$\lambda_3 = 0{,}000004437$ (Figura 28)

En las figuras que se muestran a continuación se muestra un ejemplo de los resultados calculados mediante dicho método (Figura 26, Figura 27, Figura 28). De color rojo se representa

la función representada en el intervalo mostrado mientras que las rectas de distintos colores indican la posición de la raíz.

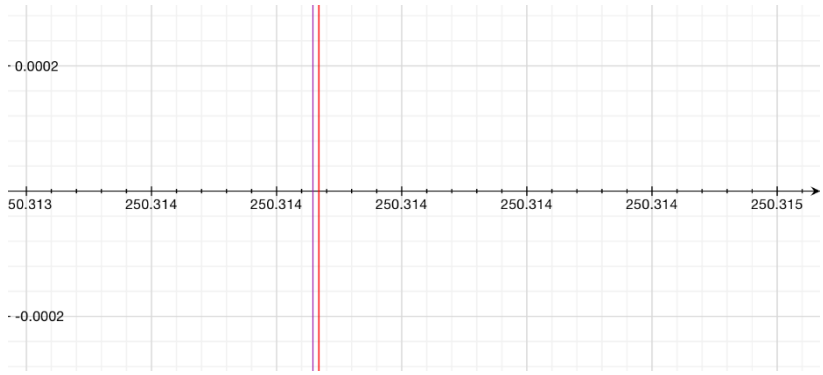

Figura 26: Estimación del primer autovalor

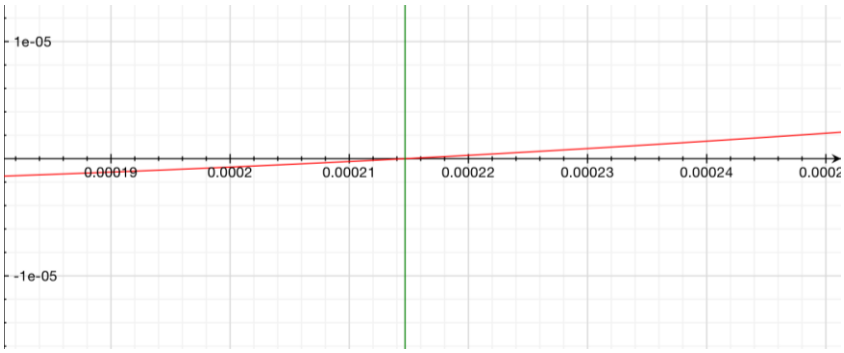

Figura 27: Estimación del segundo autovalor

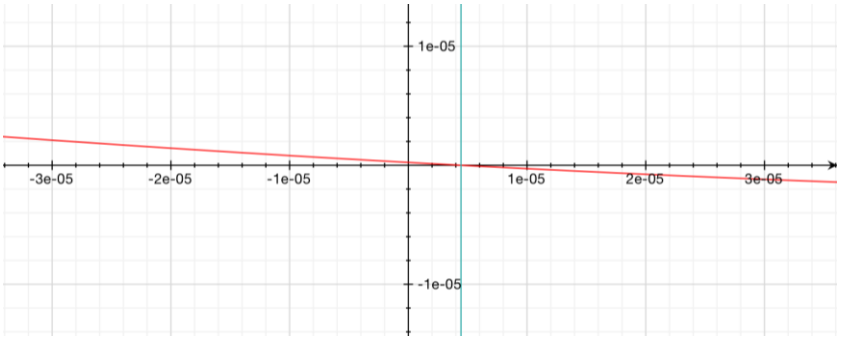

Figura 28: Estimación del tercer autovalor

4.4.3. Correlación

Con el índice de correlación se pretende medir el grado de similitud que hay entre 2 muestras, para ello, se va a programar este proceso en la GPU debido al alto número de operaciones que debe realizar en cada una de las iteraciones. En cada iteración se compara una muestra del tamaño fijado con una porción del modelo A (Figura 29) del mismo tamaño con origen en {X,Y,Z} y orientación θ. Siendo estos valores los que se modifican en cada iteración (Figura 30).

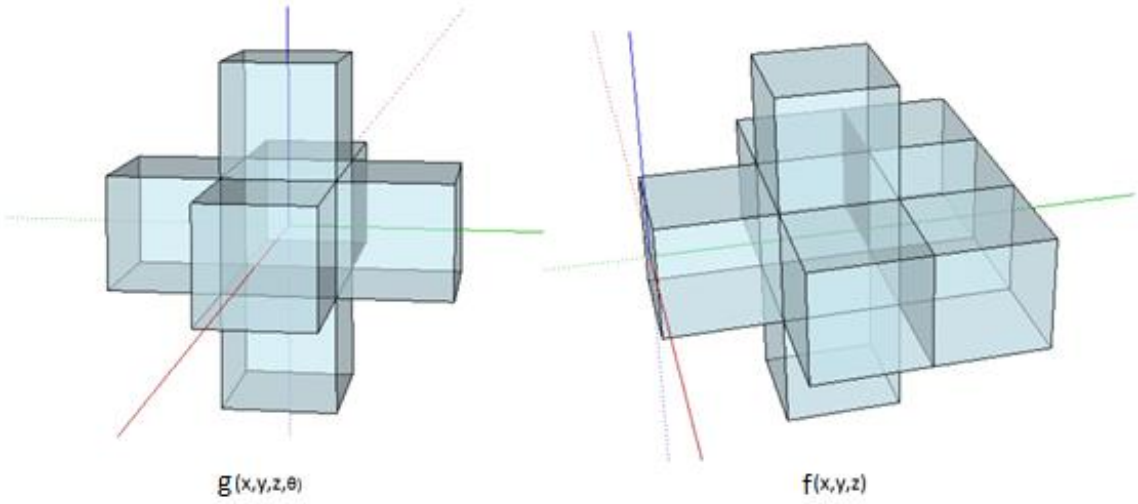

Figura 29: Estructura a localizar 'g' y estructura del mapa 'f'

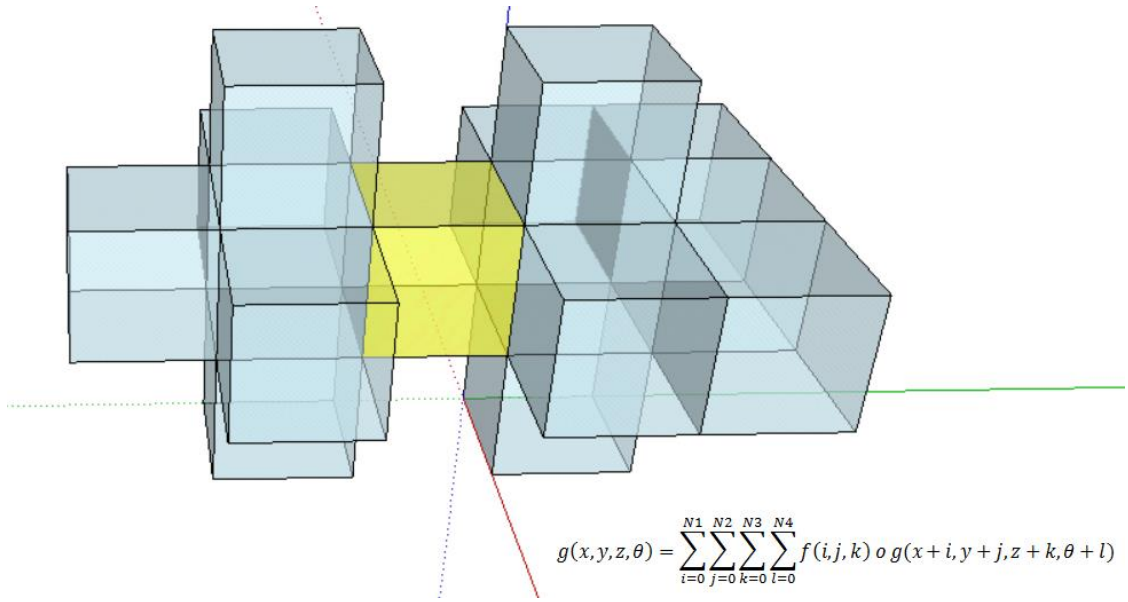

Figura 30: Etapa de correlación

Los resultados obtenidos dependen del número de emparejamientos positivos que se encuentren, pudiendo escoger el criterio que proporcione que una comparación resulte positiva. Los criterios pueden ser mediante igualdad celda ocupada-celda ocupada, igualdad celda-celda, igualdad celda-celda vecino. En la Figura 31 son representados los valores resultantes debidos a un proceso de correlación unidimensional para las estructuras mostradas en la Figura 29.

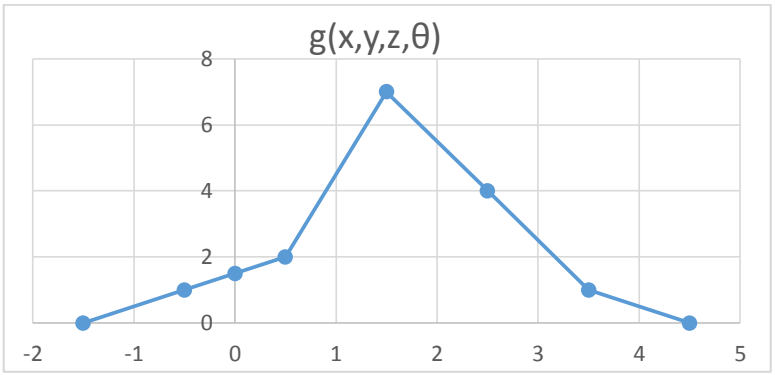

Figura 31: Función de correlación para un caso de comparación unidimensional

Para obtener los valores de la función de correlación se ha implementado una función que consiste en realizar múltiples llamadas desde la CPU a un kernel que es ejecutado en la GPU. La función consiste en un conjunto de bucles anidados, con un nivel de profundidad igual al número de variables asociadas a los grados de libertad del robot que son desconocidas, que actualizan los valores de las coordenadas, en el caso actual {X,Y,Z,θ}, y que son enviados como parámetros de entrada al kernel de la GPU. Dicho kernel, se encarga de calcular el valor de la función de correlación para cada una de las posibles localizaciones. Para el caso de que la correlación fuera en 3 dimensiones, se puede hacer una llamada al kernel con una estructura tridimensional de los threads que lo van a ejecutar, siempre que el número de threads por bloque esté permitido. A nivel de rendimiento no afecta, pero facilita las tareas de programación. En el caso de tener 4 o más ejes, se recomienda hacer uso de una estructura de threads unidimensional, ya que la dimensión máxima de la estructura de los threads es 3. La estructura del kernel consiste en el producto de ambas funciones y una operación atómica de suma para realizar el sumatorio de los valores proporcionados por cada uno de los núcleos. Para ello realiza una lectura del valor de la memoria y le suma el resultado del producto realizado anteriormente. Esta operación atómica consiste en permitir el acceso a una variable o zona de memoria únicamente a un núcleo, evitando que varios núcleos realicen una tarea de lectura o escritura sobre la misma de forma simultánea. Si en vez de realizar una operación atómica se realizara una operación de suma tradicional en la GPU el resultado sería impredecible al no ejecutarse de forma correcta el sumatorio

5. EXPERIMENTOS Y RESULTADOS

A lo largo de este capítulo se mostrarán los resultados de aquellas funciones que son de mayor relevancia para el desarrollo del método:

5.1. Modelado del entorno

Durante la realización de dicha tarea, un punto muy importante será el de la elección de los tamaños de ventana del algoritmo Block Matching para realizar el mapa de disparidad del entorno (Figura 32). Como se muestra a continuación, en la Figura 33 y Figura 34 se muestran el mapa de disparidad y la proyección de dicho mapa de disparidad a unas coordenadas tridimensionales para un tamaño de ventana pequeño. El significado de este tamaño de ventana es que obtendremos un nivel de detalle mayor, ya que se ha dividido la imagen en un mayor número de cuadrados, obteniendo un nivel de disparidad para cada uno de ellos. El problema de elegir un tamaño de ventana pequeño es el siguiente. Debido a la existencia del ruido, aquellos pixeles que se vean influidos por él adquirirán un mayor peso a la hora de realizar la etapa de emparejamiento que si la ventana tiene un mayor tamaño. El tamaño de ventana actúa como filtro paso bajo.

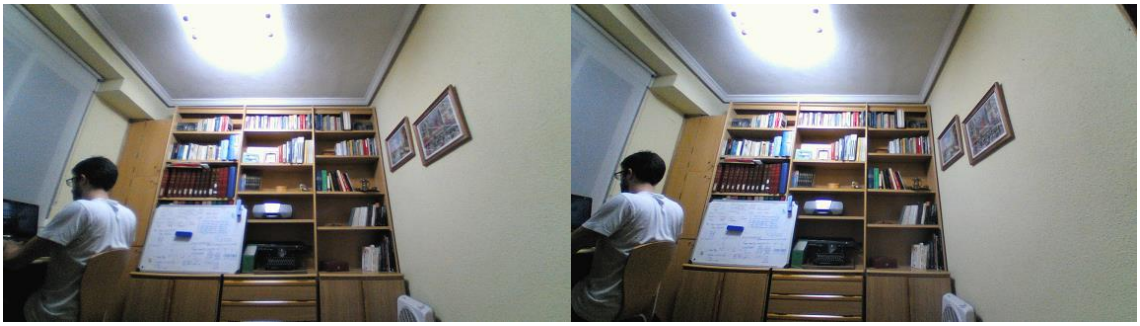

Figura 32: Imágenes izquierda y derecha del entorno

Figura 33: Mapa de disparidad visualizado con un tamaño de ventana menor

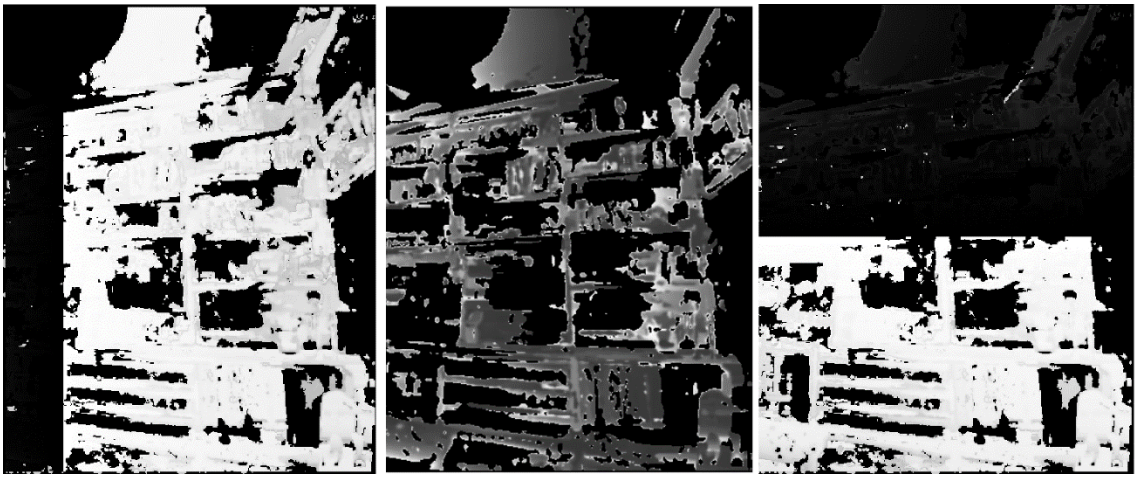

Figura 34: Componentes X Y Z de los elementos visualizados con un tamaño de ventana menor

Por el contrario, si se elige un tamaño de ventana mayor, podemos observar en la Figura 35 y Figura 36 como se ha perdido nivel de detalle, pero obteniendo unas superficies con menos ruido y más suavizadas. Esto se debe a que al disponer de unos cuadrados de mayor tamaño, se computan un mayor número de pixeles como un único conjunto, aplicándose para todos ellos un mismo valor de disparidad.

Figura 35: Mapa de disparidad con un tamaño de ventana de mayor

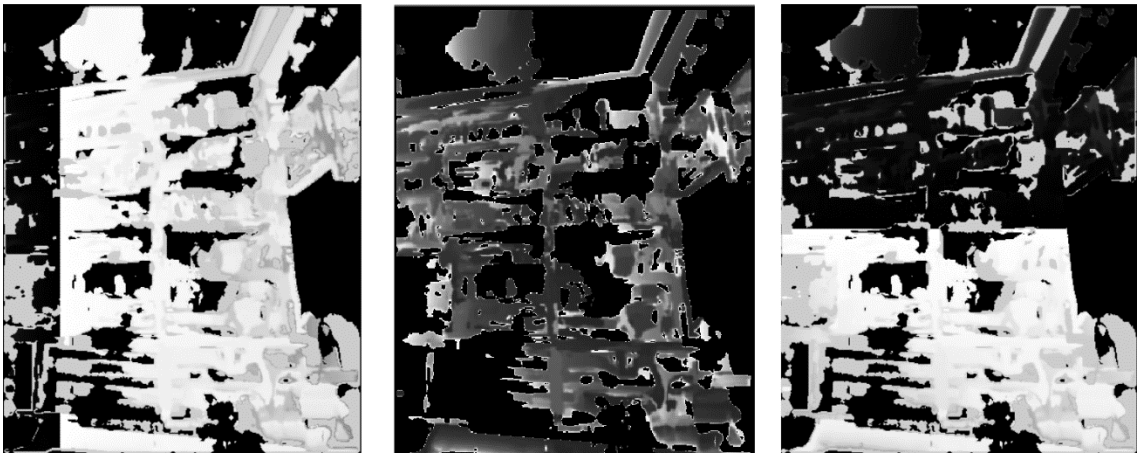

Figura 36: Componentes X, Y, Z de los elementos visualizados con un tamaño de ventana mayor

Algoritmo ICP

Como se ha explicado en la sección 4.2.2, el algoritmo ICP es un algoritmo dedicado a encontrar aquellas regiones de 2 mapas de profundidad con elementos comunes. Para ello toma 2 imágenes consecutivas y busca las transformaciones geométricas que minimizan el error cuadrático.

Iteración 1

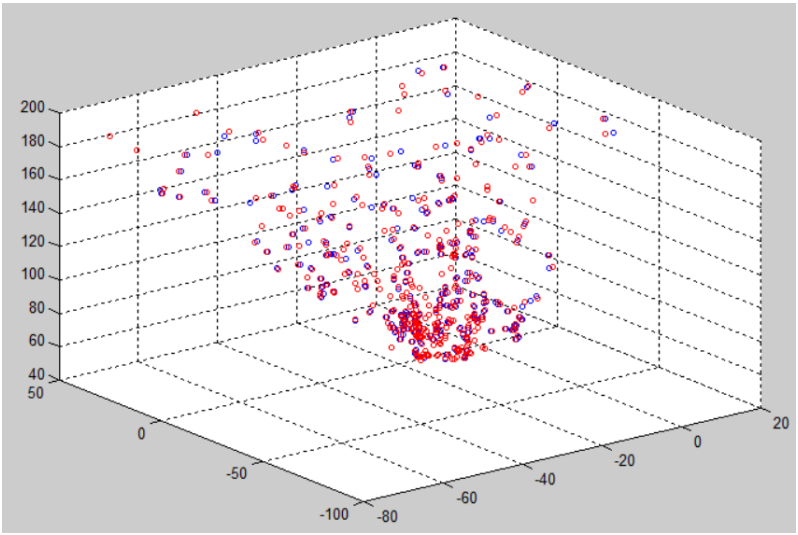

Figura 37: Primera iteración del algoritmo ICP

	Muestra (S)	Modelo (M)
Centro de gravedad X	-16,3747	-16,3506
Centro de gravedad Y	-1,7487	-1,70407
Centro de gravedad Z	93,3296	93,2868
error	0,04976	

En la primera iteración (Figura 37) se utilizan únicamente los datos de los sensores inerciales, y con los resultados obtenidos calculamos la matriz de transformación homogénea que se utilizará para el siguiente paso.

Iteración 2

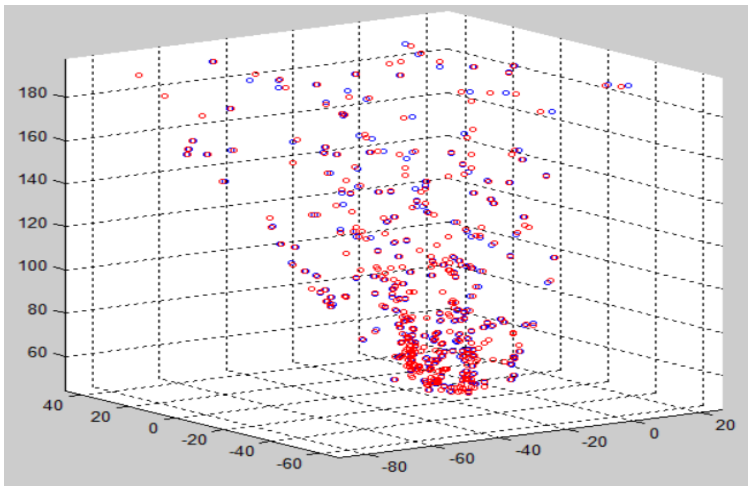

Figura 38: Iteración 2 del algoritmo ICP

	Muestra (S)	Modelo (M)
Centro de gravedad X	-16,35065	-16,37863
Centro de gravedad Y	-1,7407	-1,7421
Centro de gravedad Z	93,2869	93,2611
error	0,03738	

Con los datos de la iteración anterior se calculan los nuevos valores de error (Figura 38). Debido a que el error de la iteración 2 es inferior al de la iteración 1 y que debemos buscar el valor de la matriz de transformación homogénea que minimice el error, actualizamos la matriz de transformación homogénea a la calculada el paso anterior y se procede a calcular una nueva

Visto como es el funcionamiento, se omiten los pasos intermedios debido a que su funcionamiento es igual a los vistos en las iteraciones 1 y 2. Para explicar la condición de parada mostramos las 2 últimas iteraciones que se realizan.

Iteración N

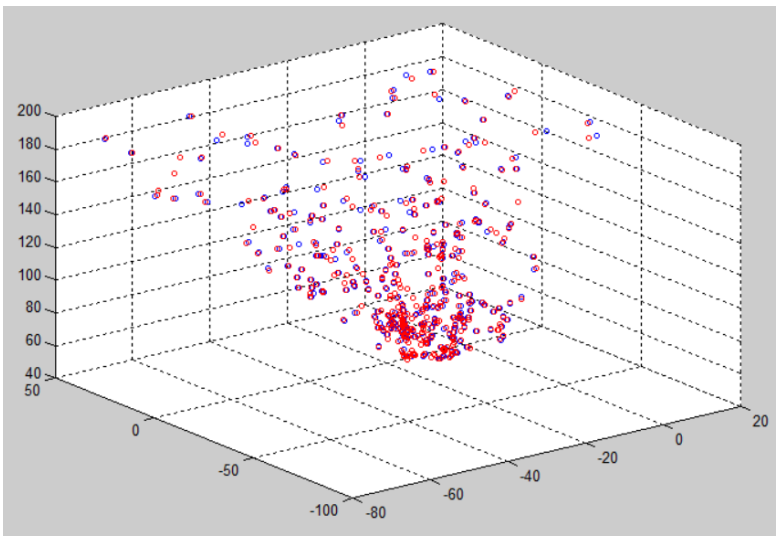

Figura 39: Iteración N del algoritmo ICP

	Muestra (S)	Modelo (M)
Centro de gravedad X	-16,4193	-16,4157
Centro de gravedad Y	-1,7342	-1,7348
Centro de gravedad Z	93,2318	93,2291
error	0,00456	

El error ha disminuido respecto de la iteración anterior (Figura 39), por lo que se realizará de nuevo el cálculo de una posible matriz de transformación homogénea mejor.

Iteración N+1

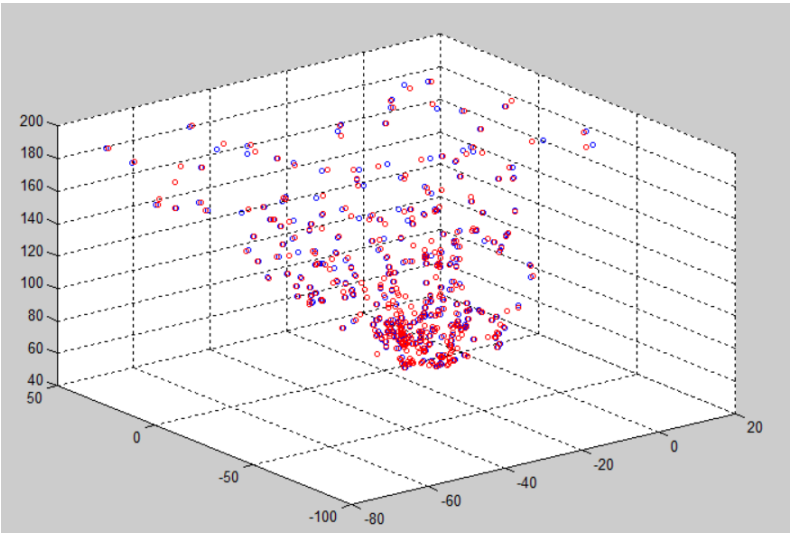

Figura 40: Iteración N+1 del algoritmo ICP

	Muestra (S)	Modelo (M)
Centro de gravedad X	-16,4157	-16,4219
Centro de gravedad Y	-1,7348	-1,7323
Centro de gravedad Z	93,2292	93,2199
error	0,0099	

Como se puede observar, el error en la iteración N+1 del algoritmo ICP es mayor que el error de la iteración N (Figura 40), por lo que nos estamos alejando del óptimo deseado. Debido a ello, se escogerá como matriz de transformación homogénea final aquella con la que se han obtenido los resultados de la iteración N. Esa será la matriz de transformación con la que se opere la muestra entera para añadir los nuevos puntos al modelo del mapa. En la siguiente gráfica (Figura 41) se observa la distribución que tiene el error, que se puede aproximar a una distribución normal con un valor de media positiva.

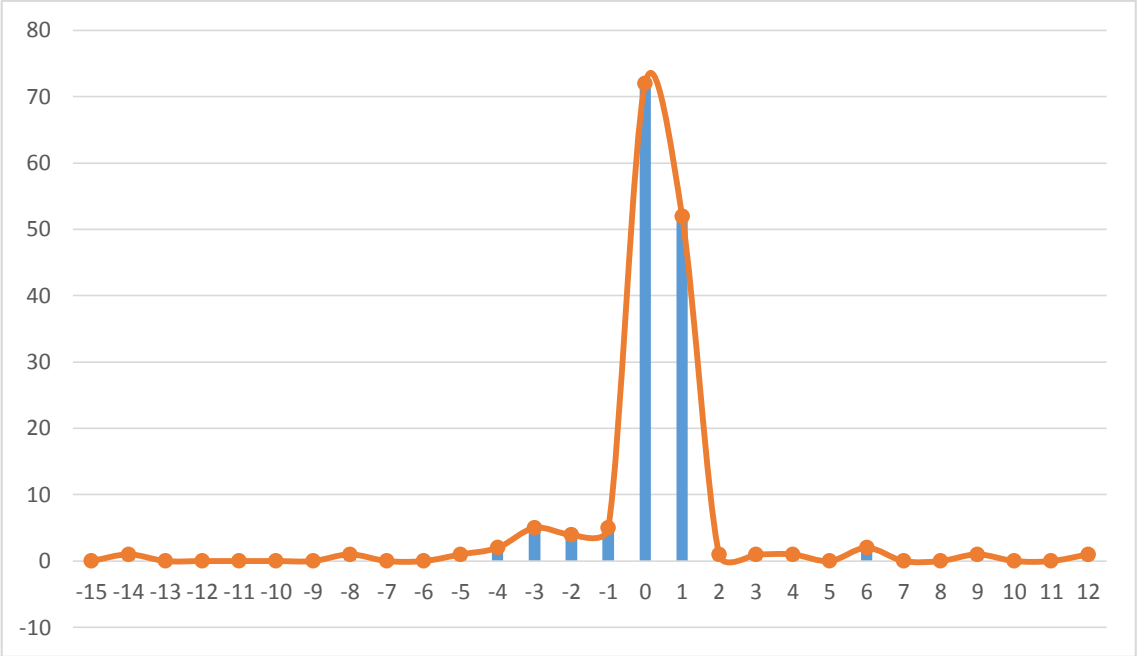

Figura 41: Distribución de la función de error

Como observación de este método comentar que los tiempos invertidos en este algoritmo son poco constantes debido a que no siempre se realizan el mismo número de iteraciones hasta converger al mínimo de la función objetivo. A continuación se muestra gráfica indicando los tiempos invertidos en dicho algoritmo para una muestra de población 50 (Figura 42).

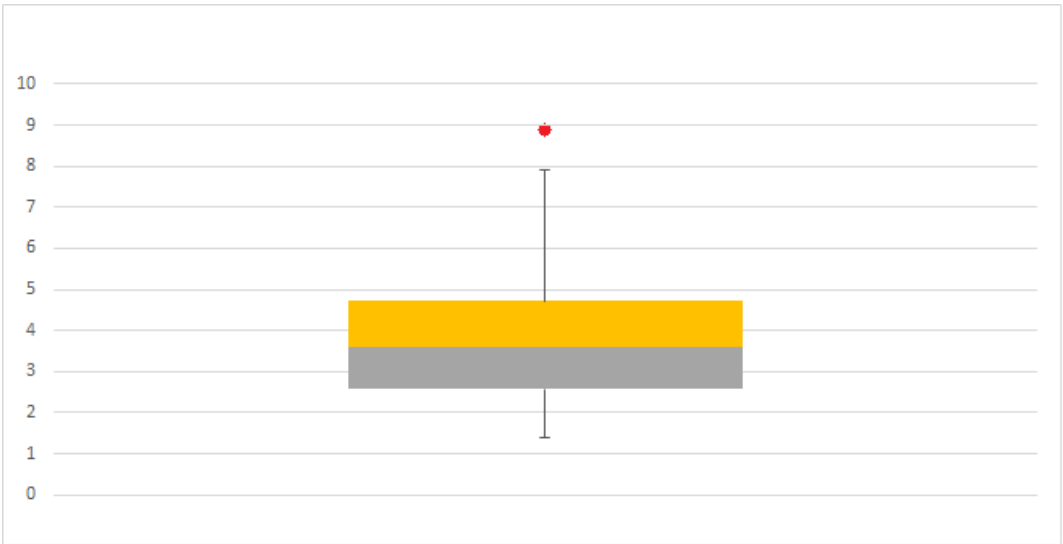

Figura 42: Diagrama Box-Plot de tiempos del algoritmo ICP

5.2. Localización

La función de localización es junto a la función encargada del algoritmo ICP la que más tiempo de procesamiento requiere, y cuanto mayor es el modelo del mapa mayor es el tiempo invertido en dicha función. Para la siguiente figura (Figura 43) se muestran los valores de la función de

correlación aplicada a un plano que incluye a la solución, los niveles de gris más claros indican un valor de correlación mayor, mientras que los oscuros uno más bajo.

Para valorar la calidad de esta función como parte del método se ha tenido en cuenta el número de posibles soluciones encontradas para un mapa con estructuras no repetitivas, que en dicho caso se observa que solamente aparece una única solución como viable y la región de incertidumbre es relativamente pequeño, siendo más preciso el método cuanto menor sea dicha región de incertidumbre. Desde un punto de vista frecuencial, la calidad se mediría como una señal impulsional, idealmente asemejado a una delta de Dirac.

La realización del ensayo se ha llevado a cabo en el entorno de la Figura 32, para comprobar la confusión que puede causar a la función de localización la repetición de elementos similares.

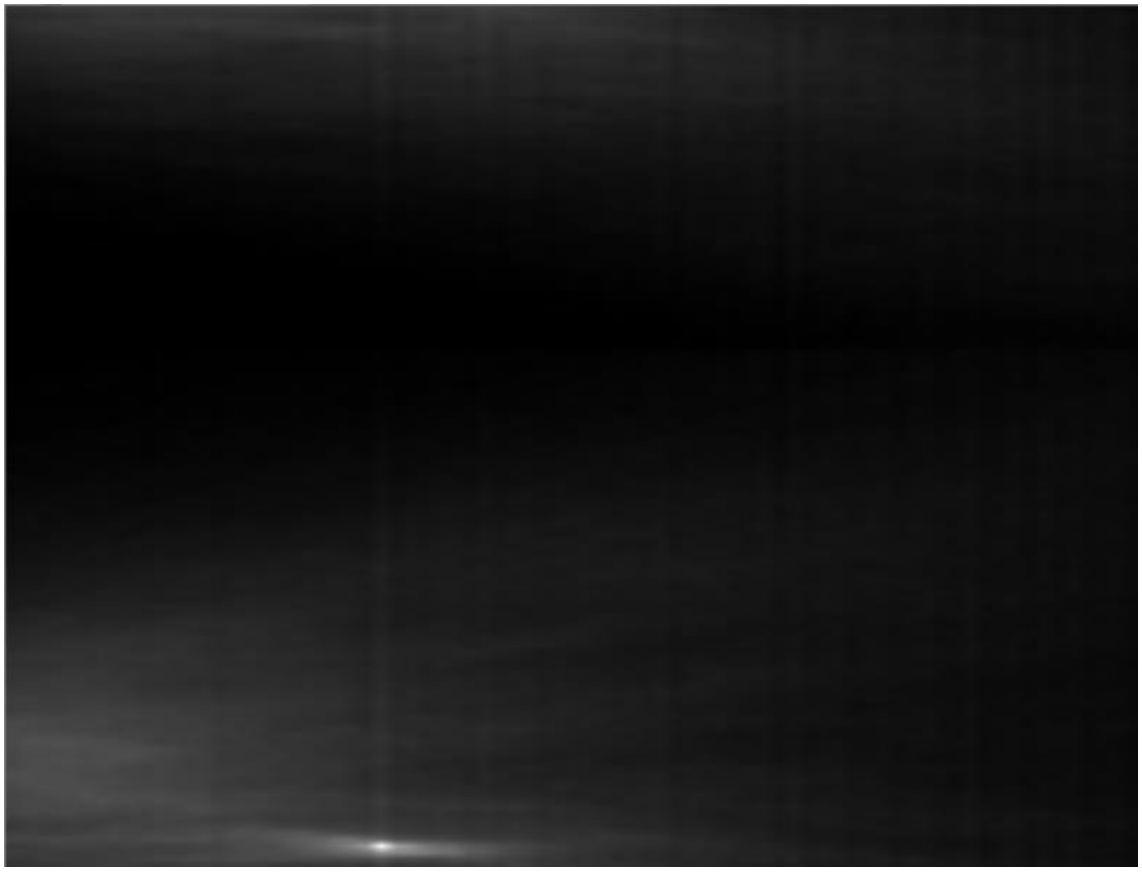

Figura 43: Función de correlación para un plano solución

6. CONCLUSIONES Y LÍNEAS FUTURAS

Finalizadas la realización de pruebas del método descrito se procede a analizar los resultados obtenidos para evaluar las ventajas e inconvenientes de su uso.

6.1. Conclusiones

En este documento se ha desarrollado un método de localización de robots mediante visión estéreo. El motivo de ello era conseguir un sistema de localización que supliera las carencias y los inconvenientes de otros sistemas ya existentes, además de que contara con unos costes menores. Como se ha podido apreciar en el capítulo 5, la medición de distancias con visión estéreo no ofrece las mismas características frente a otros sistemas, como pueden ser la resolución y la exactitud de la medida, además estos sistemas tienen dificultades para detectar superficies homogéneas. Pero como ventajas se pueden apreciar los bajos costes que tienen, tanto de instalación como de mantenimiento, las pequeñas dimensiones y el bajo peso. Estas características en ocasiones resultan decisivas en ciertas aplicaciones, por lo que la menor precisión se puede justificar en estos casos.

Para el primer paso del método, al analizar los resultados del algoritmo ICP, se llega a la conclusión de que el algoritmo utilizado consigue una gran precisión en la fusión de mapas de profundidad, obteniendo unos valores de error suficientemente bajos como para que en la etapa de localización sean poco influyentes. Pero para que este algoritmo sea realmente efectivo, es necesario de realizar primeramente una aproximación de las coordenadas de los puntos, para evitar que el algoritmo obtenga como solución óptima un mínimo local en vez de obtener el mínimo global de la función de coste.

El motivo por el que los errores del algoritmo ICP se han considerado despreciables es que durante el paso de localización se construye una maya de celdas, cada una de esas celdas tiene un tamaño superior al tamaño de dichos errores, por lo que esta acción ayudará a reducir la influencia de los errores. El principal problema encontrado es que se trata de un error acumulativo, al estar referenciando los desplazamientos de una imagen frente a la anterior, aunque este problema no es exclusivo de este método, sino que está en todos aquellos que únicamente se basen en sus sensores exteroceptivos para estimar la posición sin ningún soporte adicional.

En el paso de localización, los resultados de la búsqueda son positivos, se ha comprobado que es un método robusto y fiable, ya que como se puede observar en la Figura 43, la función de correlación discrimina aquellos localizaciones encontrando el origen desde el cual han sido tomadas las imágenes, pero con el inconveniente de requerir mucho tiempo de procesado, por lo que no es viable para operaciones en tiempo real. Como una forma de agilizar los tiempos de localización, se opta por incluir la brújula de 3 ejes junto al resto de sensores. Este sensor reduce de manera significativa los tiempos y el coste de implantación es muy bajo.

Este método ha sido desarrollado para resolver el problema de SLAM de localización global, apartado 2.4.5, pero limitando el entorno de búsqueda, puede utilizarse también como solución al problema de seguimiento de posición. El método cuenta con la ventaja frente a otros sistemas de localización de no tener errores aditivos, hablando únicamente del paso de

localización, ya que los errores de las iteraciones anteriores no repercuten al estar comparándose frente al modelo.

6.2. Líneas futuras

Una posible mejora de dicho método debería estar relacionada con la forma de crear las mallas de celdas. En la actual implementación, se dedican muchas de dichas celdas a representar espacios vacíos que no aportan ninguna información al método. Esto se debe a que en el entorno por el cual el robot se desplaza es mayoritariamente espacio vacío, y por consiguiente va a aumentar mucho la cantidad de memoria consumida y el tiempo de procesado en el proceso.

Por ello como mejora se propone realizar un método que en vez de representar un único volumen, limitado por los planos de las paredes, techo y suelo, se extraigan los distintos planos y que se cree un volumen sobre ellos que permita acoger los datos procedentes del relieve de estos. Dichos planos deberán almacenar la información necesaria para saber cuáles son sus dimensiones y los planos con los que interseca. Con esta implementación se lograría reducir la cantidad de memoria almacenada y en gran medida el número de iteraciones que desde un principio se podrían descartar debido a la ausencia de información, agilizando el tiempo de localización.

Otra mejora que se le puede realizar está enfocada a mejorar la calidad del mapa. Se realizaría en una etapa intermedia entre los pasos 1 y 2 en la que es aplicado un suavizado de formas. Con ellos se pretende eliminar ciertos errores debidos al proceso de generación de los mapas de profundidad, además de utilizar técnicas probabilísticas para mejorar el algoritmo ICP. Otra posibilidad sería la creación del modelo mediante un software CAD. Esto aumentaría los tiempos de modelado para entornos grandes y con gran cantidad de detalles, pero sin embargo se eliminan los errores que proporcionan los instrumentos de medida que son acumulativos, mejorando el paso de localización posterior.

BIBLIOGRAFÍA

[1] E. D. Kaplan y C. Hegarty, Understanding GPS: principles and applications, Artech house, 2005.

[2] G. Pajares Martinsanz, Visión por computador: imágenes digitales y aplicaciones, Ra-Ma, 2001.

[3] J. A. Blackburn, Modern Instrumentation for scientists and engineers, Springer Science & Business Media, 2001.

[4] F. A. Candelas Herías y J. A. Corrales Ramón, *Fusión GypsyGyro-Ubisense,* 2007.

[5] H. J. Arditty y H. H. C. Lefevre, «Sagnac effect in fiber gyroscopes,» *Optics letters,* vol. 6, nº 8, pp. 401-403, 1981.

[6] A. P. Aguiar y J. P. Hespanha, «Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty,» *IEEE Transactions on Automatic Control,* vol. 52, nº 8, pp. 1362-1379, 2007.

[7] Y. Cheng, M. W. Maimone y L. Matthies, «Visual odometry on the Mars exploration rovers-a tool to ensure accurate driving and science imaging,» *Robotics & Automation Magazine, IEEE,* vol. 13, nº 2, pp. 54-62, 2006.

[8] A. Ciprián Chico, Diseño y desarrollo de un sistema de posicionamiento en interiores basado en Wi-Fi con tecnologia Android, Universidad Carlos III, 2009.

[9] S. Thrun, W. Burgard y D. Fox, Probabilistic Robotics, The MIT Press, 1999.

[10] P. Henry, M. Krainin, E. Herbst, X. R. Ren y D. Fox, «RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments,» de *12th International Symposium on Experimental Robotics (ISER)*, 2010.

[11] K. Karimi, N. G. Dickson y F. Hamze, «A performance comparison of CUDA and OpenCL,» *arXiv,* nº arXiv:1005.2581, 2010.

[12] A. d. l. Escalera Hueso, Visión por computador : fundamentos y métodos, Prentice Hall, 2001.

[13] L. Yih-Chuan y T. Shen-Chuan, «Fast Full-Search Block-Matching Algorithm for Motion-Compensated Video Compression,,» *IEEE Transactions on Communications,* vol. 45, nº 5, 1995.

[14] J.-N. Kim y T.-S. Choi, «A fast three-step search algorithm with minimum checking points using unimodal error surface assumption,» *IEEE Transaction on Consumer Electronics,* vol. 44, nº 3, pp. 638-648, 1998.

[15] G. Ferrer Mínguez, Integración Kalman de sensores inerciales INS con GPS en un UAV, Universitat Politècnica de Catalunya, 2009.

[16] A. Baena Alonso, P. Martín Lerones y J. Gómez García-Bermejo, Alineamiento de imagenes tridimensionales, Universidad de Valladolid.

ANEXO A: Presupuestos

Se adjunta en este capítulo el presupuesto de este proyecto. En la Figura 44 se muestran detalles sobre el proyecto como son la duración del proyecto y el presupuesto dedicado al mismo. En la Figura 45 se realiza un desglose de los costes del proyecto, justificando así el gasto del importe del presupuesto. Y por último, en la Figura 46, se muestra el resumen de dichos costes.

1.- Autor:				
Miguel Díez-Ochoa Díez				
2.- Departamento:				
Departamento de Sistemas y Automática				
3.- Descripción del Proyecto:				
- Titulo	**Localización de robots móviles en entornos tridimensionales**			
	mediante visión estéreo y CUDA			
- Duración (meses)	**6**			
Tasa de costes Indirectos:		**21%**		
4.- Presupuesto total del Proyecto (valores en Euros):				
	Euros	29.751,65		

Figura 44: Descripción del proyecto

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) a)	Coste hombre mes	Coste (Euro)
Díez-Ochoa Díez Miguel		Ingeniero	9	2.694,39	24.249,51
		Ingeniero Senior		4.289,54	0,00
		Ingeniero		2.694,39	0,00
					0,00
					0,00
		Hombres mes	**9**	**Total**	24.249,51

a) 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable d)
WideCam F100	34,04	100	6	60	3,40
WideCam F100	34,04	100	6	60	3,40
Arduino Mega 2560	39,00	100	30	60	19,50
GY-85: Sensor inercial 9 ejes	7,25	100	6	60	0,73
Placa ProtoBoard	4,75	100	60	60	4,75
Ordenador portatil	969,00	100	19	60	306,85
				Total	338,63

Figura 45: Desglose presupuestario

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	24.249,51
Amortización	338,63
Subcontratación de tareas	0,00
Costes de funcionamiento	0,00
Costes Indirectos	5.163,51
Total	29.751,65

Figura 46: Resumen de costes del proyecto

ANEXO B: Estructura de una nube de puntos para un formato de archivo PCD

Para que los resultados obtenidos durante el modelado del entorno puedan ser visualizados, se incluye a continuación la estructura que utilizan las librerías PCL (Point Cloud Library) para almacenar los valores de los elementos de las nubes de puntos. En la realización de este proyecto hemos incluido únicamente las componentes {X,Y,Z} de los elementos, sin incluir ninguna referencia al color de cada uno de ellos.

A continuación se muestra una porción de un archivo PCD.

http://pointclouds.org/documentation/tutorials/pcd_file_format.php

```
# .PCD v.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 213
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 213
DATA ascii
0.93773 0.33763 0 4.2108e+06
0.90805 0.35641 0 4.2108e+06
0.81915 0.32 0 4.2108e+06
0.97192 0.278 0 4.2108e+06
0.944 0.29474 0 4.2108e+06
0.98111 0.24247 0 4.2108e+06
0.93655 0.26143 0 4.2108e+06
0.91631 0.27442 0 4.2108e+06
0.81921 0.29315 0 4.2108e+06
0.90701 0.24109 0 4.2108e+06
0.83239 0.23398 0 4.2108e+06
0.99185 0.2116 0 4.2108e+06
```

Los campos que se incluyen en la cabecera del archivo son los siguientes:

- **VERSION**: indica la version del archive PCD.
- **FIELDS:** especifica la representación de los elementos del archivo.
- **SIZE**: especifica el tamaño en bytes de cada una de los tipos.
- **TYPE**: especifica el tipo de dato, pudiendo ser integer, float,o tipo unsigned.
- **COUNT**: especifica el número de elementos que tienes cada una de las dimensiones
- **WIDTH**: indica la anchura del conjunto de datos de la nube de puntos.
- **HEIGHT**: indica la altura del conjunto de datos de la nube de puntos:

- **VIEWPOINT**: especifica el vector de translación y los quaternios del origen de adquisición de los elementos.
- **POINTS**: indica el número total de elementos del archivo.
- **DATA**: indica si el archive es del tipo ASCII o binario

ANEXO C: prueba de rendimiento del equipo utilizado con CUDA Y OpenCL

Comparación de rendimientos entre CUDA y OpenCL en el equipo utilizado

	CUDA	OpenCL
Rendimiento agregado de shader	190.5MPixel/s	179.08MPixel/s
Nativo punto flotante shaders	736MPixel/s	679.17MPixel/s
Nativo punto flotante FP64 shaders	49.3MPixel/s	47.22MPixel/s
Nativo Quad Shaders	2MPixel/s	1.54MPixel/s
Precisión de Punto-Flotante (Normal - FP32)		
Análisis Científico Agregado	46.38GFLOPS	44.92GFLOPS
Multiplicar la Matriz Genérica (GEMM)	48.65GFLOPS	49.49GFLOPS
Transformada Rápida de Fourier (FFT)	44.22GFLOPS	40.77GFLOPS
Simulación N-Body	268.19GFLOPS	235.54GFLOPS
Precisión de Punto-Flotante (Alta - FP64)		
Análisis Científico Agregado	18.78GFLOPS	16.37GFLOPS
Multiplicar la Matriz Genérica (GEMM)	25.3GFLOPS	23.88GFLOPS
Transformada Rápida de Fourier (FFT)	14GFLOPS	11.22GFLOPS
Simulación N-Body	23.66GFLOPS	21.45GFLOPS
Rendimiento agregado de la memoria	17.68GB/s	17.68GB/s
Ancho de banda de memoria interna	51.74GB/s	51.63GB/s
Ancho de banda de transferencia de datos	6GB/s	6GB/s
Tiempos de Benchmark		
Capacidad de Tiempo para Copiar	39µs	39µs
Capacidad de Tiempo para Leer	324µs	323µs
Capacidad de Tiempo para Escribir	338µs	338µs
Anomalía de la prueba de rendimiento		
Ancho de banda de memoria interna	51.74GB/s	51.63GB/s
La eficiencia de ancho de banda	66.25%	66.11%
Sistema a ancho de banda del dispositivo	5.92GB/s	5.92GB/s
La eficiencia de ancho de banda	94.64%	94.64%
Dispositivo a ancho de banda del sistema	6.18GB/s	6.2GB/s
La eficiencia de ancho de banda	98.81%	99.09%
SiSoftware Sandra Lite (Evaluacion)		
2014.03.20.21		